DOCUMENT RESUME

ED 054 126 SP 007 324

AUTHOR	Hannigan, Joseph T.; Engvall, Richard E.
TITLE	Introduction to Computer Programming.
INSTITUTION	Framingham Public Schools, Mass.
PUB DATE	70
NOTE	127p.
EDRS PRICE	EDRS Price MF-$0.65 HC Not Available from EDRS.
DESCRIPTORS	*Computer Science Education, *Curriculum Guides, *Programing

ABSTRACT

GRADES OR AGES: No mention. SUBJECT MATTER: Computer programing. ORGANIZATION AND PHYSICAL APPEARANCE: The guide is divided into seven chapters, each of which is in outline form with numerous diagrams and charts. It is mimeographed and looseleaf-bound with a paper cover. OBJECTIVES AND ACTIVITIES: No objectives are mentioned. Each of the seven chapters--covering history of computers, numeration systems, flow charting, equipment, machine and assembly language, basic, and focal--consists of a detailed content outline followed by several problems to be assigned. Problems involve performing calculations or writing programs. INSTRUCTIONAL MATERIALS: Several references are listed at the beginning of each chapter. STUDENT ASSESSMENT: No mention. [Not available in hardcopy due to marginal legibility of original document.] (RT)

ED054126

~~PROCESS WITH MICROFICHE AND PUBLISHER'S PRICES.~~
MICROFICHE REPRODUCTION ONLY.

INTRODUCTION TO COMPUTER PROGRAMMING

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

Summer

1970

Prepared by

Joseph T. Hannigan
Richard E. Engvall

Framingham Public Schools
Framingham, Massachusetts
01701

Albert L. Benson
Superintendent of Schools

George P. King
Associate Superintendent

Edward J. Moran
Director of Secondary Instruction

SP007324

TABLE OF CONTENTS

CHAPTERS

1. History of Computers
2. Numeration Systems
3. Flow Charting
4. Equipment
5. Machine Language and Assembly Language
6. Fortran
7. Basic
8. Focal

APPENDICES

I. Reference Books

II. Sample Programs
   a. Machine and Assembly Language
   b. Fortran
   c. Basic
   d. Focal

III. Programmer's Reference Forms
   a. Machine Language and Assembly Language
   b. Fortran
   c. Basic
   d. Focal

IV. Teletype Keyboard and Directions for Use
   a. Preparation of Tapes

V. Diagnostics
   a. Basic
   b. Focal
   c. Fortran

VI. Assignments
   a. Basic
   b. Focal
   c. Fortran
   d. Machine Language

PREFACE

INTRODUCTION TO COMPUTER PROGRAMMING is a new course. It is based on the belief that a computer can be an effective tool in teaching. The student who understands the basic concepts of a computer may then use the computer to aid his learning. This course is intended to provide insight into the computer, and into the concepts involved in programming.

There is very little documented experience in teaching a course of this type. It requires extensive preparation by the teacher. He must be competent in operating the computer to be used by the students. He must understand and appreciate the algorithmic approach to solving problems. He must be willing to give freely of his time.

There are four general objectives of INTRODUCTION TO COMPUTER PROGRAMMING. The first is to provide a solid foundation of knowledge of computers and what they can do. The second is to teach students to use a computer to lessen the computational burden of mathematical and scientific investigation. The third is to reinforce concepts taught in mathematics and science classes. The fourth is to provide the student the opportunity to extend himself to the limits of his own creativity, to investigate topics unrelated to classroom material, and to develop the ability to work independently.

This manual is provided to help you teach or learn this course. It is not complete, nor will it ever be complete. Each year revisions, deletions and additions will be made. Sample programs will be inserted where they are deemed helpful. More reference books will be listed as they become known. The chapter on FORTRAN will be included. Suggestions for teaching the course will be included. After the course has been taught, the deficiencies in the manual will be more apparent and will be corrected. Until then, the teacher and student are encouraged to make frequent use of the suggested reference materials.

It should be noted that the entire manual need not be taught. Suggested alternatives are:

I. Chapters 1, 2, 3, 4, 5, 6.
II. Chapters 1, 3, 6, 7.
III. Chapters 1, 2, 3, 4, 5, 7.
IV. Chapters 1, 3, 6, 7, 8.
V. Chapters 1, 2, 3, 4, 5, 8.

Problems in the body of the manual should be assigned when they occur. Problems in Appendix VI may be assigned at the teacher's option.

CHAPTER 1

HISTORY

AN OUTLINE HISTORY OF COMPUTERS

I. Counting Tools

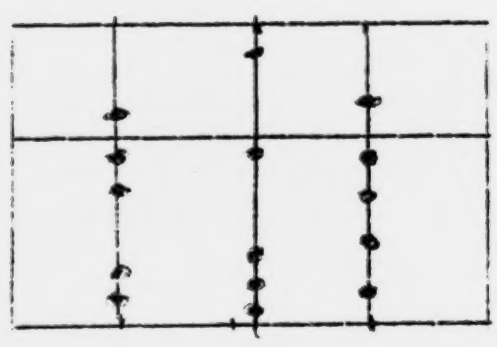
23,000 B. C. - Historians tell us that man began to count by using a one to one correspondence without the use of any number names. What man probably did do was to associate one stone with one animal when the animals were let out to pasture. When the animals returned, he would subtract one stone for each animal. If any stones were left, he knew that all the animals had not returned.

18,000 B. C. - It is about this time that man developed the concept of "one", "two", "many".

3,000 B. C. - Man developed the Abacus.

Chinese Abacus - "Suan-Pan"

Japanese Abacus - "Soro-Ban"



Aside:

In 1947; an American named Tom Wood had a contest with a Japanese named Kiyoshi Matsuzake to see who could calculate (correctly) the fastest. Mr. Wood used an electric calculating machine while Mr. Matsuzake competed on the Soro-Ban. Mr. Matsuzake won in four of the five problem categories.

300 B. C. - The first recorded use of zero for place value.

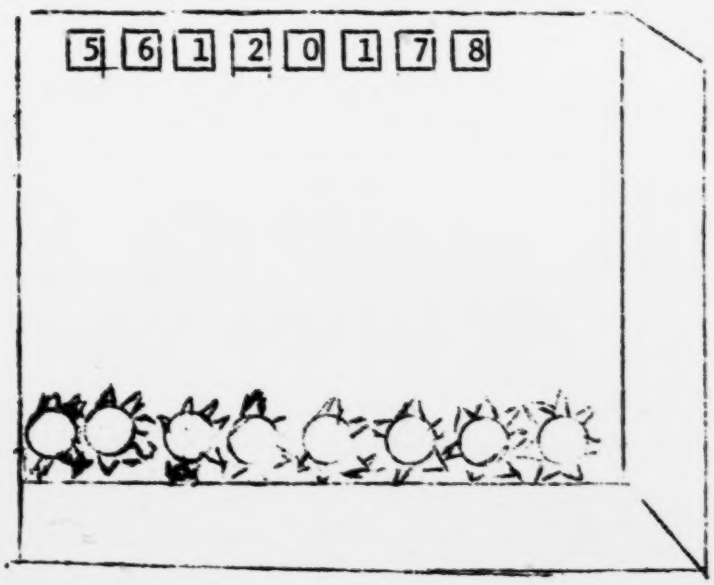
1,200 A. D. - The Arabic numerical system came into being.

II. Mechanical Devices and more recent computers

1617 - logarithms developed by John Napier. A device known as "Napier's Bones" peformed multiplications using logarithms.

1621 - slide rule developed by William Oughtred (first analog computer)

1642 - first adding machine - built by Blaise Pascal (at age 19) he worked in his father's tax office - built a gear driven computer that would add 8 columns of numbers.



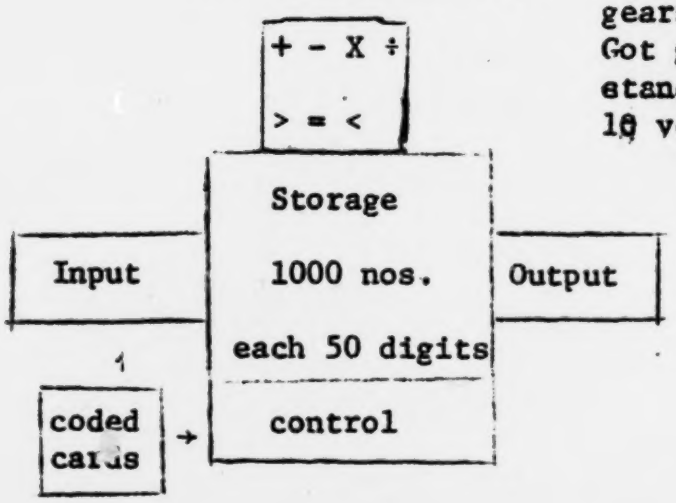
aside → also fathered projective geometry, hydrodynamics, and probability

1674 - calculating machine which performed multiplication developed by Leibnitz

III. Punched Card Devices

1801 - Joseph Jacquard - used punched cards in weaving in his automatic loom the punched holes supplied instructions that controlled the selection of threads and the application of designs.
To show off, Jacquard had these cards weave his portrait - took 20,000 cards.

1812 Charles Babbage - designed "Difference Engine" - hundreds of gears and shafts, ratchets and counters. Got grant of 17,000 ($1,000,000 by today's standards) (5 decimal place accuracy) worked 10 years - gave up designed "Analytical Engine"



Analytical Engine Features
- Memory Unit
- Punched cards for Input and Control
- 20 place accuracy
- Arithmetic Unit
- Automatic Readout

Went broke - never finished - would have taken over an acre to house the machine

1887 - Dr. Herman Hollerith - Bureau of Census - still working on 1880 census (50 million people)
Size of card corresponded to 1887 U. S. dollar bill
1890 census (62 million) took only until 1892
System would:
- Record
- Sort
- Tally
- Print

Hollerith used electricity - cards floated across pools of mercury - pins dropping through holes made contact creating an electrical circuit causing counters to add one. Start of IBM after several mergers.
50-75 cards/min

IV. Modern Computors

1925 - Vannevar Bush (MIT) built analog computer

1949 - Howard Aiken (Harvard) - built MARK I
used relays — 3 operations per second
punched paper tape — first of large scale
electromechanical — digital computers

1946 - Mauchly & Eckert (U of Penn) ENIAC - Electronic Numerical Integrator & Computer
1st electronic computer - 18000 vacumn tubes
weighed 30 tons
1500 Sq. Ft. (walls as long as football field)
5,000 operations/second

1949 EDSAC - Cambridge, England - 1st stored program computer

1951 - MIT WHIRLWIND - used magnetic core memory

1951 - UNIVAC I (Mauchley & Eckert) - first commercially available computer.

1st Generation
Vacumn Tubes — Bulky, took much room, produced much heat

2nd Generation (1960) - solid state devices (transistors)
- smaller
- less heat
- more reliable

3rd Generation (1964)

microminiaturized circuits (integrated circuits)
thin film memory
smaller
faster (op speed - billionths of a second)
(manosecond)

CHAPTER 2

NUMERATION SYSTEMS

NUMERATION SYSTEMS

I. To Convert Numbers From One Base to Another Base.

A. To convert Numbers in the Decimal System to the Binary System.

1. The Highest Power Method

Change 37 to Base Two

1) The highest power of two that is in 37 is $2^5 = 32$
Divide 37 by 32
32 $\overline{)37}$ with a remainder of 5
1

2) Take the 5 and divide it by the next lowest power of 2 which is $2^4 = 16$ getting 0 with a remainder of five

3) Divide the remainder (5) by the next lowest power of two $2^3 = 8$ getting zero with 5 as a remainder

4) Repeat this process until the power's of two are exhausted. (i.e. until the last division is one)

5) Your answer is the number formed by reading down the quotients column.

Example

	Divisors	Remainders	Quotients	Read Down
1)	32	37	1	
2)	16	5	0	↓
3)	8	5	0	
	4	5	1	
4)	2	1	0	
	1	1	1	

5) Ans: $37_{10} = 100101_2$

2. The Remainder Method

1. Divide 37 by 2 getting 18 with a remainder of one. Place the 18 under the 37 and the 1 to the right of the 37.
2. Next divide the 18 by 2 getting 9 with a remainder of zero. Place the 9 below the 18 and the zero to the right of 18.
3. Keep dividing by 2 until you get a quotient of zero. You then read up the remainder column writing your answer left to right as you read up the column.

	Divisors	Quotients	Remainders
1)	2	37	1
2)	2	18	0
	2	9	1
	2	4	0
3)	2	2	0
	2	1	1 Read UP
		0	

Ans: $37_{10}=100101_2$

3. **Change Binary Numbers to Decimals (base ten)**

Set up in table form as illustrated with the top row containing the base to be changed written in exponential form. The second row contains the values of the top row in base ten. The last row contains the number to be changed in its proper column.

1. Take the binary number 10110_2 and place these numbers in the first five columns from the right.

2. Find the products of the second row and the third row and find their sum.

Example

Exponential Form	2^4	2^3	2^2	2^1	2^0
Value	16	8	4	2	1
(1)	1	0	1	1	0_2

(2)

$16 \times 1 = 16$
$8 \times 0 = 0$
$4 \times 1 = 4$
$2 \times 1 = 2$
$1 \times 0 = 0$
22

Ans. $10110_2 = 22_{10}$

NUMERATION SYSTEMS - PROBLEM SET

I. Convert the following decimal numbers to their binary equivalent.

1) 15
2) 18
3) 42
4) 100
5) 235
6) 1
7) 294
8) 117
9) 86
10) 4090
11) 4978
12) 1502
13) 7777
14) 501
15) 828
16) 907
17) 15631
18) 6023
19) 18741
20) 99999
21) 10101010
22) 15712

II. Convert the following binary numbers to their decimal equivalents.

1) 110_2
2) 101_2
3) 1110110_2
4) 1011110_2
5) 0110110_2
6) 11111_2
7) 1010_2
8) 110111_2
9) 11011011101_2
10) 111000111001_2
11) 1110101101001_2
12) 11111111011101_2
13) 10101101010010_2
14) 111111_2
15) 000101001_2
16) 111111111111_2

B. Convert numbers in the decimal svstem to the octal svstem.

1. The Highest Power Method

Change 127 to base eight.

1. Find the highest power of eight that goes into 127. It is $8^2 = 64$. Divide 127 bv 64 giving 1 with a remainder of 63. Place the 1 to the right of the 127 and the 63 below the 127.

2. Divide the 63 bv the next lowest power of 8 ($8^1 = 8$). This gives an answer of 7 with a remainder of 7.

3. Divide the 7 by the next lowest power of 8 which is $8^0 = 1$ giving an answer of 7 with a remainder of zero. Place the 7 to the right of the 7 and the zero below the 7.

4. You read your answer reading down the right hand column and writing the answer from left to right as you read.

Example

				Read Down
1)	64	127	1	↓
2)	8	63	7	
3)	1	7	7	
		0		

4) Ans. $127_{10} = 177_8$

2. The Remainder Method

1. Divide 127 bv 8 getting 15 with a remainder of 7. Place the 15 below the 127 and the 7 to the right of 127.

2. Divide 15 bv 8 getting 1 with a remainder of 7. Place the 7 to the right of the 15 and the 1 below the 15.

3. Divide the 1 bv 8 getting zero and a remainder of 1. Place the zero below the 1 and the 1 to right of the 1.

4. You find vour answer bv reading up the right hand and writing the numbers from left to right as vou read.

Example

			Remainders	
1)	8	127	7	
2)	8	15	7	
3)	8	1	1	↑ Read Up
		0		

4) Ans. $127_{10} - 177_8$

---

3. Change from the Octal System to the Decimal System.

Set up in table form as illustrated with the top row containing the base to be changed written in exponential form. The second row contains the values of the top row in base ten. The last row contains the number to be changed with each digit place in its proper column.

1. Take the octal number 3745_8 and place these numbers in the first four columns from the right.

2. Find the products of the second row and the third row and find their sum.

Example

Exponential Form	8^5	8^4	8^3	8^2	8^1	8^0
Value	32768	4096	512	64	8	1
1)			3	7	4	5_8

2) 512 x 3 = 1536
64 x 7 = 448
8 x 4 = 32
1 x 5 = 5
2021

Ans.: $3745_8 = 2021_{10}$

Problems: Convert the following decimal numbers to their octal equivalents.

1)	796	7)	108	13)	15072	18)	287
2)	32	8)	999	14)	107891	19)	19872
3)	4037	9)	57	15)	571	20)	4279
4)	580	10)	111	16)	8572		
5)	1000	11)	97	17)	7852		
6)	3	12)	4095				

Convert the following octal numbers to their decimal equivalents.

1)	67_8	7)	777_8	13)	64_8	19)	25634_8
2)	701_8	8)	106_8	14)	117_8	20)	124563_8
3)	32_8	9)	20_8	15)	245_8		
4)	236_8	10)	171_8	16)	271_8		
5)	10_8	11)	576_8	17)	171_8		
6)	71_8	12)	75_8	18)	662_8		

C. Convert from binary to octal

In order to convert from binary to octal you must understand the following table which gives the comparative values between the binary system and the octal system.

Base Two	Base Eight
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

---

1. To convert from binary to octal, you will group the binary number into groups of three and then find the equivalent values for each group in base eight.

Take the binary number, separate it into groups of threes, working from right to left. The equivalent values in base eight are written next which gives you your answer in base eight.

$10\ 001\ 011\ 110\ 101\ 111_2$

$\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$

$2\ 1\ 3\ 6\ 5\ 7_8$

$10001011110101111_2 = 213657_8$

---

2. To convert from Octal to binary, you still use the tables given earlier. You convert your octal number to its equivalent in the binary system. For each octal number there are three binary digits. It is the reverse process of working from the binary to the octal.

Take an octal number and convert each digit into its binary equivalent which gives you your answer $37246_8 = 011111010100110_2$

$3\ 7\ 2\ 4\ 6_8$

$\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$

$011\ 111\ 010\ 100\ 110_2$

Problems: Convert the following binary numbers to their octal equivalents.

1) 101_2
2) 111_2
3) 1101_2
4) 10111_2
5) 1101_2
6) 11101_2
7) 1011_2
8) 1001_2
9) 1111_2
10) 1011_2
11) 10110_2
12) 11101_2
13) 10001_2
14) 11001_2
15) 11011_2
16) 1010_2
17) 1100_2
18) 10111_2
19) 110001_2
20) 110011_2
21) 111000_2
22) 1110011_2
23) 101100111_2
24) 1111001110_2
25) 1011000_2
26) 10101010_2
27) 111001111_2
28) 101111011_2
29) 110011100_2
30) 1011101111_2
31) 10001_2
32) 1100111_2
33) 11111111_2
34) 11110101_2
35) 1110001110001_2
36) 111100011_2
37) 11111100001011_2
38) 110110110_2
39) 101111001_2
40) 1101111101101_2
41) 111111111111_2
42) 111011011111_2
43) 101010101010_2
44) 110111011110_2
45) 1110111011111_2
46) 11111011110_2
47) 11101110110001_2
48) 1101110110110010_2
49) 10111011_2
50) 11111101000111001_2

Convert the following octal numbers to their binary equivalents.

1) 354_8
2) 736_8
3) 15_8
4) 10_8
5) 7_8
6) 307_8
7) 70_8
8) 12_8
9) 121_8
10) 310_8
11) 111_8
12) 100_8
13) 1101_8
14) 64_8
15) 3214_8
16) 414_8
17) 707_8
18) 141_8
19) 747_8
20 1243_8
21 1001_8
22) 1221_8
23) 401_8
24) 273_8
25) 2211_8
26) 6713_8
27) 5142_8
28) 347_8
29) 2768_8
30) 1347_8
31) 2653_8
32) 1347_8
33) 4375_8
34) 6754_8
35) 72313_8

Convert the following octal numbers to their binary equivalents. (cont.)

36) 5276_8

37) 6114_8

38) 54377_8

39) 342156_8

40) 101101_8

41) 27531_8

42) 72367_8

43) 111111_8

44) 1011101_8

45) 274362_8

46) 57643_8

47) 765_8

48) 1100001_8

49) 220022_8

50) 33303_8

II. ADDITION IN BASE TWO AND IN BASE EIGHT

Addition in all bases is exactlv the same as vou have alwavs done in base ten. You onlv have to remember when to carrv from one column to the next. When you add one to nine in base ten, you know nine is the largest single digit number in base ten so you now carrv one over into the ten's column giving ten. This same thing happens in other bases except the largest single digit varies with the base being used. The largest single digit in anv base is the digit that is one less than the base being used. In base eight the largest single digit is seven. In base five the largest single digit is four. In base eight when you add one to seven you get 10_8

I think we can best illustrate this thought bv some examples. In base eight let's add 276_8 and 322_8

$$\begin{array}{r} {\scriptstyle 11} \\ 276_8 \\ \underline{322_8} \\ 620_8 \end{array}$$

Six and two added gives 10_8 so we carrv one into the eights column. Add the carried one and seven we get 10_8 so we carrv the one over to the next column. We now add the zero and the two in the second column and get two. We add the one and the two and the three in the third column and get a sum of 6.

Let us now look at an addition problem in base two. The largest digit in base two is one less than two which is one. Adding one to one in base two gives 10_2 which is a two digit number.

Let us look at an example to illustrate addition in base two. Let's add 1011_2 to 111_2

$$\begin{array}{r} {\scriptstyle 1111} \\ 1011_2 \\ \underline{111_2} \\ 10010_2 \end{array}$$

Adding one and one in the first column gives 10_2. Carrv the one over the next column and write zero in vour answer. Now add the carried one to one getting 10_2 which means we have to carrv one again. Add the zero and the one in the second column getting one which vou write in vour answer. In the third column add the carried one to the one alreadv in the column getting 10_2 which means carrving one to the fourth column. Add the carried one to the given one getting 10_2 so vou have to carrv one to the next column This is the onlv number in this column so vou place the one in vour answer.

Problems: Add the folowing problems in base two.

1) $1011_2 + 101_2$

2) $1111_2 + 111_2$

3) $11110_2 + 1011_2$

4) $11011_2 + 11_2$

5) $1011011_2 + 111_2$

6) $110110011_2 + 110111_2$

7) $1110011100_2 + 11100111_2$

8) $11110111_2 + 1110111_2$

9) $1110_2 + 11001_2$

10) $11111_2 + 1111111_2$

11) $1101101111_2 + 1111110111_2$

12) $1111001110_2 + 11011_2$

13) $1111001110_2 + 110111_2$

14) $1101111000_2 + 10101011_2$

15) $1101110011111_2 + 10101010_2$

16) $111110011_2 + 11011011_2$

17) $1111_2 + 10110110_2$

18) $111111_2 + 10001000_2$

19) $100011000_2 + 111001111_2$

20) $1110111100_2 + 101011_2$

Find the sum of the following problems in base eight.

1) $24_8 + 13_8$

2) $32_8 + 14_8$

3) $152_8 + 313_8$

4) $416_8 + 231_8$

5) $2314_8 + 3252_8$

6) $534_8 + 253_8$

7) $4356_8 + 311_8$

8) $374_8 + 413_8$

9) $1753_8 + 6027_8$

10) $47631_8 + 1457_8$

11) $53712_8 + 4726_8$

12) $75246_8 + 2562_8$

13) $13756_8 + 47221_8$

14) $147312_8 + 5723671_8$

15) $62357_8 + 214673_8$

III. A Subtraction of Numbers in other bases.

Let's take a look at subtraction as you know it and then we shall learn how to subtract by addition.

Let's subtract 321 from 476.

The standard form for this is
$$\begin{array}{r} 476 \\ -\ 321 \\ \hline 155 \end{array}$$
which gives us a difference of

Now as you will see in section B, you can get the same answer except that instead of subtracting one number from another, you will add.

III. B. Subtraction by Addition

In order to perform subtraction by addition, you have to know how to find the 10's compliments of the subtrahends. The 10's complements of a number is that number which when added to the number gives a higher power of ten.

Example: The complement of 9 is 1 because 9 + 1 = 10; the complement of 9 is 91 because 9 + 91 = 100; etc. The ten and the one-hundred are powers of ten.

Subtraction by addition using the ten's complement.

A) 476 → 476 Add the numbers
− 321 → + 679 ← the 10's complement
(1)155

Discard

The answer is 155

Subtract 321 from 1000, there must be an equal number of digits in both numbers.

B) 3476 → 3476 Add the numbers
− 243 → +9757 ← The ten's complement
(1)3233

Discard

The answer is 3233

Another method of subtraction by addition is by using the nine's complement. You simply take one less than the ten's complement to get the nine's complement.

Subtraction by addition using the nine's complement.

A)

```
  476    →   476    (Add the numbers
 -321        +678   +(Nine's complement (Subtract 321 from 999)
          (1)154
           \- +1     Bring the one around and ADD
        Answer 155
```

B)

```
 3476    →  3476     (Add the numbers
 - 243      +9756   + (Nine's complement (Subtract 243 from 999,
          (1)3232                        remember that there must be an
             +1     Bring the one around  equal number of digits in
        Answer 3233 and ADD.              both numbers.)
```

Problems: Find the Difference bv Using the Ten's Complement and the nine's Complement.

1) 27 - 3

2) 163 - 79

3) 3479 - 268

4) 111 - 10

5) 1534 - 13

6) 725 - 696

7) 39743 - 872

8) 5783 - 10

9) 273 - 194

10) 32591 - 9237

Subtraction in Octal by Addition.

The eights complement is the number that is added to a giving number to get a power of eight.

Example: The 8's complement of 6 is 2 because $6 + 2 = 10_8$;
Subtraction bv Addition using the eights complement.

A) $746_8 \rightarrow 746_8$ Add
$-512_8 \quad +266_8$ ←Eight's complement (subtract 512 from 1000_8)
$(1)234_8$

Discard The Answer is 234_8

B) $521_8 \rightarrow 521_8$ Add
$-76_8 \quad +702_8$ 8's complement (Subtract 76 from 1000_8)
$(1)423_8$

Discard The Answer 423_8

Another method of subtraction bv addition is using the seven's complement. The seven's complement is one less than the eight's complement.

Subtraction bv Addition using the seven's complement.

A) $746_8 \rightarrow 746_8$ Add
$-512_8 \quad +265_8$ ← 7's complement (Subtract 512_8 from 777)
$(1)233_8$
$+1$ Bring around and Add
234_8

The answer 234_8

B) $521_8 \rightarrow 521_8$ Add
$-76_8 \quad +701_8$ ← 7's complement (Subtract 71_8 from 777)
$(1)422_8$
$+1$ Bring around and Add
423_8 The Answer is 423_8

Problems: Find the Difference by using the eight's complement and the seven's complement.

1)	$26_8 - 17_8$	21)	$6752_8 - 374_8$
2)	$31_8 - 4$	22)	$14325_8 - 2741_8$
3)	$171_8 - 7$	23)	$37614_8 - 4303_8$
4)	$12_8 - 5$	24)	$5104_8 - 21_8$
5)	$734_8 - 23_8$	25)	$7602_8 - 7_8$
6)	$454_8 - 142_8$	26)	$4317_8 - 206_8$
7)	$471_8 - 50_8$	27)	$2765_8 - 776_8$
8)	$675_8 - 2$	28)	$5612_8 - 23_8$
9)	$517_8 - 27_8$	29)	$23421_8 - 4321_8$
10)	$1642_8 - 531_8$	30)	$4756_8 - 714_8$
11)	$47_8 - 24_8$	31)	$62735_8 - 3046_8$
12)	$7452_8 - 147_8$	32)	$2576_8 - 67_8$
13)	$2534_8 - 1423_8$	33)	$4721_8 - 32_8$
14)	$376_8 - 12_8$	34)	$3516_8 - 600_8$
15)	$427_8 - 16_8$	35)	$4271_8 - 3$
16)	$5361_8 - 423_8$	36)	$7531_8 - 246_8$
17)	$376_8 - 107_8$	37)	$67542_8 - 57_8$
18)	$27_8 - 7_8$	38)	$1342_8 - 463_8$
19)	$6531_8 - 24_8$	39)	$7651_8 - 674_8$
20)	$7567_8 - 147_8$	40)	$4357_8 - 463_8$

Subtraction in binary by addition.

The two's complement is the number that is added to a given number to get a power of two.

Example: The two's complement of 1 is 1 because $1 + 1 = 10_2$; the two's complement of 101_2 is 11_2 because $101_2 + 11_2 = 1000_2$; etc.

Subtraction by addition using the two's complement.

A)

$1101_2 \rightarrow 1101_2$ Add

$-1011_2 \quad +0101_2 \leftarrow$ Two's complement (Subtract 1011_2 from 10000_2)

$(1)0010_2$

Discard

The answer is 10_2

B)

$10011_2 \rightarrow 10011_2$ Add

$- 101_2 \quad +1011_2 \leftarrow$ Two's complement (Subtract 101_2 from 10000_2

$(1)\ 1110_2$

Discard

Instead of using the two's complement, you can also get the same results by using the one's complement in your subtraction by addition. The one's complement is one less than the two's complement.

A)

$1101_2 \rightarrow 1101_2$ Add

$-1011_2 \quad +0100_2 \leftarrow$ One's complement (Subtract 1011_2 from 1111_2)

$(1)\ 0001_2$

$+1$ Bring around and Add.

10_2 The answer is 10_2

B)

$10011_2 \rightarrow 10011_2$ Add

$- 101_2 \quad +1010_2 \leftarrow$ one's complement (subtract 101_2 from 1111_2

$(1)\ 1101_2$

$+1$ Bring around and add

1110_2

The answer is 1110_2

Problems: Find the difference by using the two;s complement and the one's complement.

1) $101_2 - 10_2$

2) $110_2 - 1_2$

3) $1101_2 - 10_2$

4) $111_2 - 10_2$

5) $101_2 - 11_2$

6) $1011_2 - 1_2$

7) $1011_2 - 11_2$

8) $10101_2 - 1$

9) $101100_2 - 1011_2$

10) $101010_2 - 11011_2$

11) $101011_2 - 10110_2$

12) $11111_2 - 1000_2$

13) $100100_2 - 10010_2$

14) $101110_2 - 10001_2$

15) $1010110_2 - 10110_2$

16) $11001110_2 - 1001110_2$

17) $11100110_2 - 11_2$

18) $11111_2 - 1001_2$

19) $101111_2 - 10001_2$

20) $110011_2 - 11111_2$

21) $1110001100_2 - 110011_2$

22) $101100_2 - 10011_2$

23) $10111_2 - 1001_2$

24) $10111100_2 - 110000_2$

25) $11011100_2 - 101_2$

26) $11111000_2 - 1000000_2$

27) $1111000111_2 - 111111110_2$

28) $11011_2 - 10_2$

29) $110001_2 - 101_2$

30) $10110111_2 - 1011_2$

31) $10110_2 - 1_2$

32) $111000_2 - 101_2$

33) $10101_2 - 10_2$

34) $110101_2 - 1$

35) $1110110_2 - 101001_2$

36) $11100111_2 - 11000_2$

37) $101100111_2 - 101_2$

38) $1110011_2 - 101_2$

39) $1100111_2 - 110001_2$

40) $10101010_2 - 10101_2$

ANSWERS - NUMERATION SYSTEMS PROBLEM SETS

I.

1) 1111 2) 10010 3) 101010 4) 1100110 5) 11101011 6) 1

7) 100100110 8) 1110101 9) 1010110 10) 111111111010

11) 1001101110010 12) 10111011110 13) 1111001100001 14) 111110101

15) 1100111100 16) 1110001011 17) 11110100001111 18) 1011110000111

19) 100100100110101 20) 11000011010011111 21) 100110100010000100010010

22) 11110101100000

II.

1) 6 2) 5 3) 118 4) 94 5) 54 6) 31 7) 10 8) 55 9) 1757

10) 3641 11) 7529 12) 16349 13) 44372 14) 63 15) 41 16) 4095

III.

1) 16 2) 6 3) 7 4) 575 5) 676 6) 40 7) 307 8) 30

9) 277 10) 7777 11) 2653 12) 7652 13) 24132101 14) 255

15) 6444 16) 2372

IV.

1) 011001100 2) 111011110 3) 01101 4) 001000 5) 111

6) 101100010100 7) 011000111 8) 001001000001 9) 11000 10) 110100

11) 111111111111 12) 011010001100 13) 111111110110 14) 001010100011

V.

1) 1434 2) 40 3) 7705 4) 1104 5) 1750 6) 3 7) 154 8) 1747

9) 71 10) 157 11) 141 12) 7777

VI.

1) 010 000 100 2) 101 000 101 3) 001 011 011 010

VII.

	Binary	1's complement	2's complement
1)	000 000 000 001	111 111 111 110	111 111 111 111
2)	000 000 000 010	111 111 111 101	111 111 111 110
3)	000 000 000 111	111 111 111 000	111 111 111 001
4)	000 010 000 001	111 101 111 110	111 101 111 111
5)	000 011 100 001	111 100 011 110	111 100 011 111
6)	can't be done	can't be done	

VIII

1) .1001 2) .8125 3) .1101 4) .625 5) .10001 6) .110001

IV. 1) 1370, 1370 2) 3114, 3114 3) 3722, 3722 4) 7776, 7776
5) 0047, 0050 6) 2057, 2057

CHAPTER 3

FLOW CHARTS

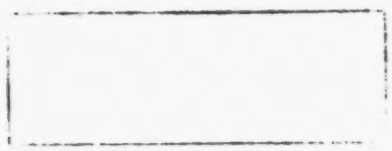
FLOW CHART

Terminal Svmbol

Process Svmbol

Decision Svmbol

Connection Svmbol



Annotation Svmbol

The flow charts or logical diagrams are devices which have received considerable attention during the past several vears as an aid to problem solving. This device is actuallv a list of steps to be performed in sequence. For claritv, the steps mav be listed within squares, circles, or other geometric figures connected bv arrows which designate the flow, or direction, of the process. For each problem that is to be solved with a digital computer it is necessarv to work out in advance definite procedures which the machine can follow without further need for supervision. This step-bv-step process is often called a program, and one wav of presenting the steps in a program is bv means of the flow chart.

Flow charts are applicable not onlv to problems that are solved with computers. Before a person solves anv problem, whether it is mathematical, how to repair an automobile, or how to bake a pie - a certain amount of planning should be done.

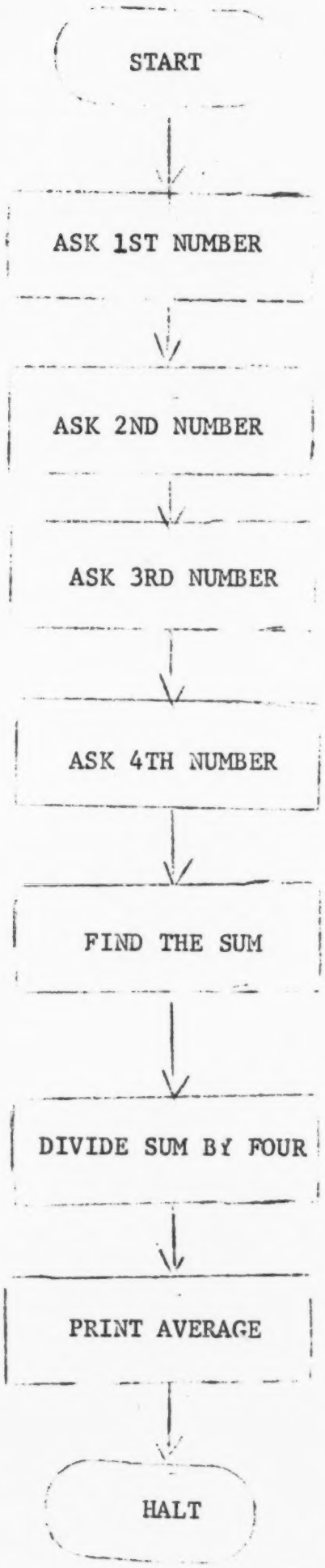
It is important that some operations be performed in a certain sequence, whereas others mav be done in anv order. For example, in changing a tire one should consider whether the car should be lifted before the lugs which hold the wheel are removed. Likewise, in making a pie one should consider whether the crust should be prepared before the oven is turned on.

There are some techniques available to help a person organize his thoughts in beginning to solve a problem. One wav is to prepare a list of things that need to be done. This list will give the order to be followed. For example, in a recipe book,. the list of directions for making a pie is merelv a wav of organizing the approach to a problem. On the other hand, the directions for solving manv problems mav reauire considering the possibilitv that a certain step cannot be done and that something else must be tried before continuing.

There are three tvpes of flow charts that are to be considered - the straightline, the loop, and the branch.

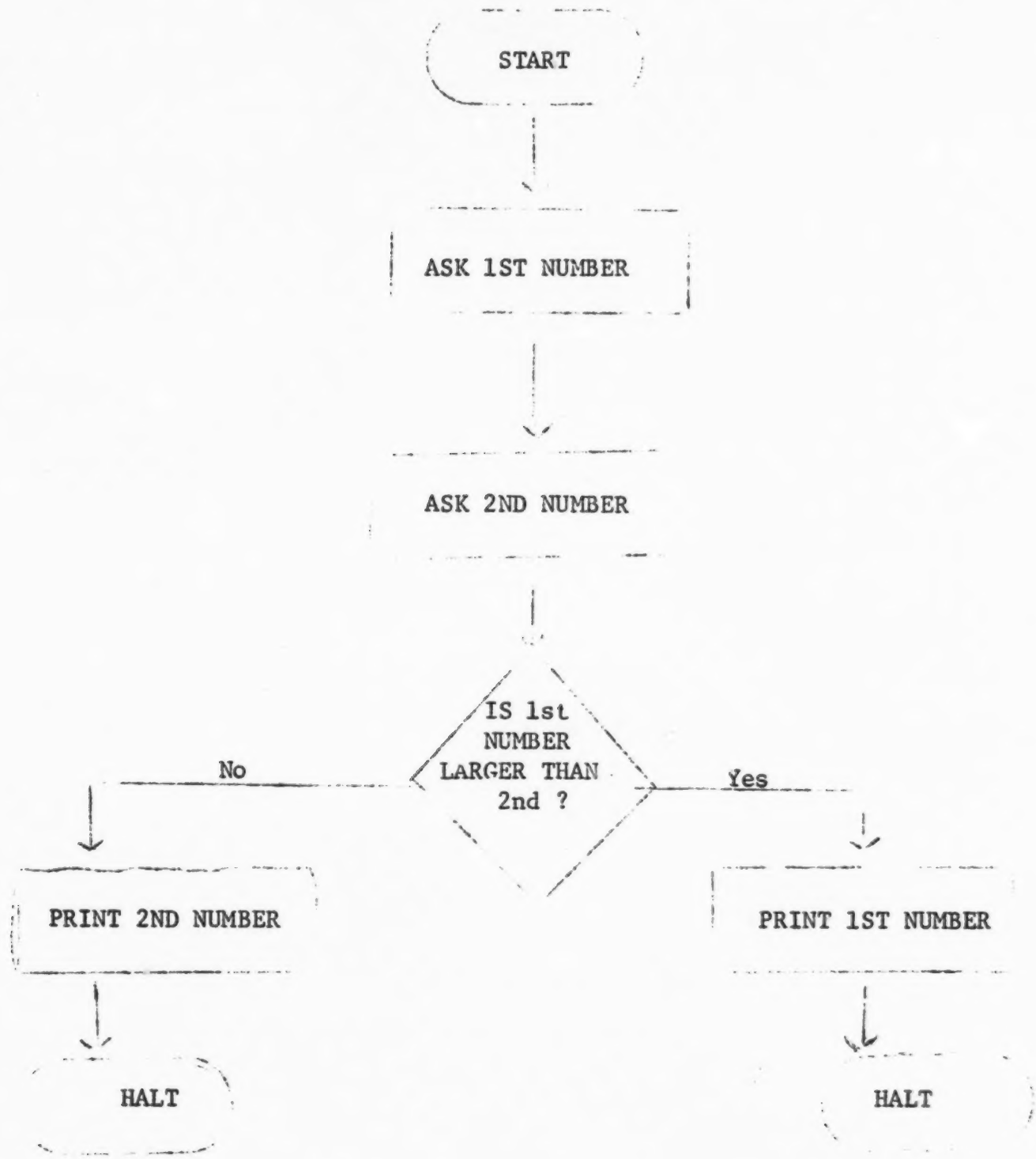
The straight line flow chart goes from one step to the next with no decisions acting on the flow chart.

An example of a straight line flow chart problem is: Find the average of four numbers.


START
ASK 1ST NUMBER
ASK 2ND NUMBER
ASK 3RD NUMBER
ASK 4TH NUMBER
FIND THE SUM
DIVIDE SUM BY FOUR
PRINT AVERAGE
HALT

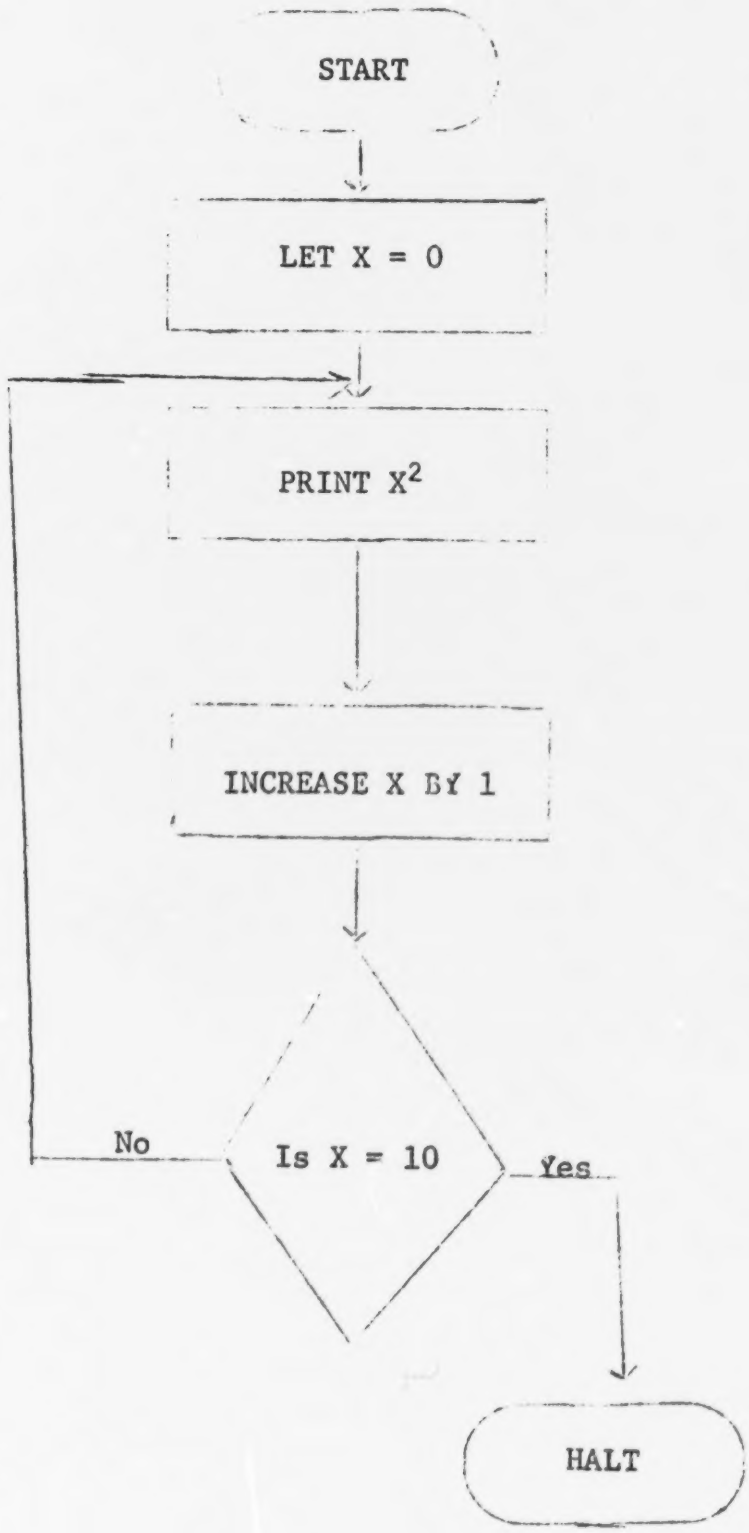
The branch flow chart will start out as a straight line flow chart, but at some point a decision will be asked. You now have a choice to make. You make the decision and you branch off in that direction to completion of the problem.

An example of a branch flow chart would be: Write the larger of unequal numbers.

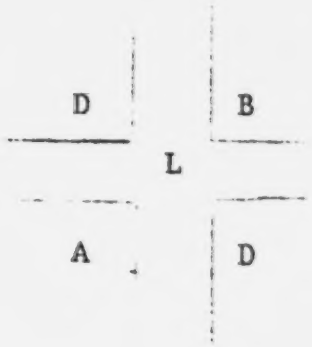


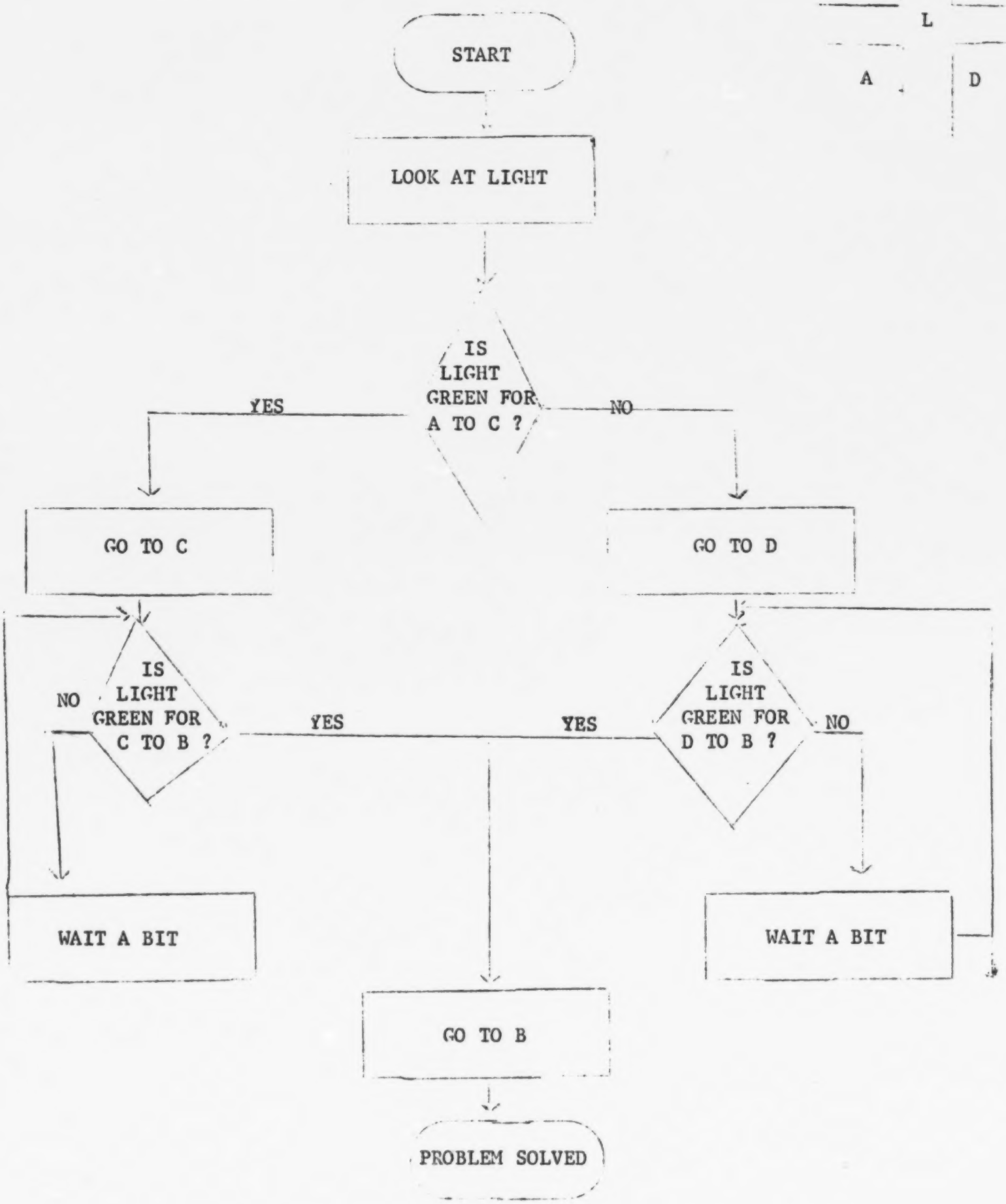
The third tvpe of flow chart is the loop. Here the flow chart goes along as a straight line flow chart until eventuallv a decision has to be made. One decision can have vou go on until completion while the other decision will have vou go back to perform some part of the program again.

An example of a loop flow chart problem would be:
Print the first ten perfect squares.

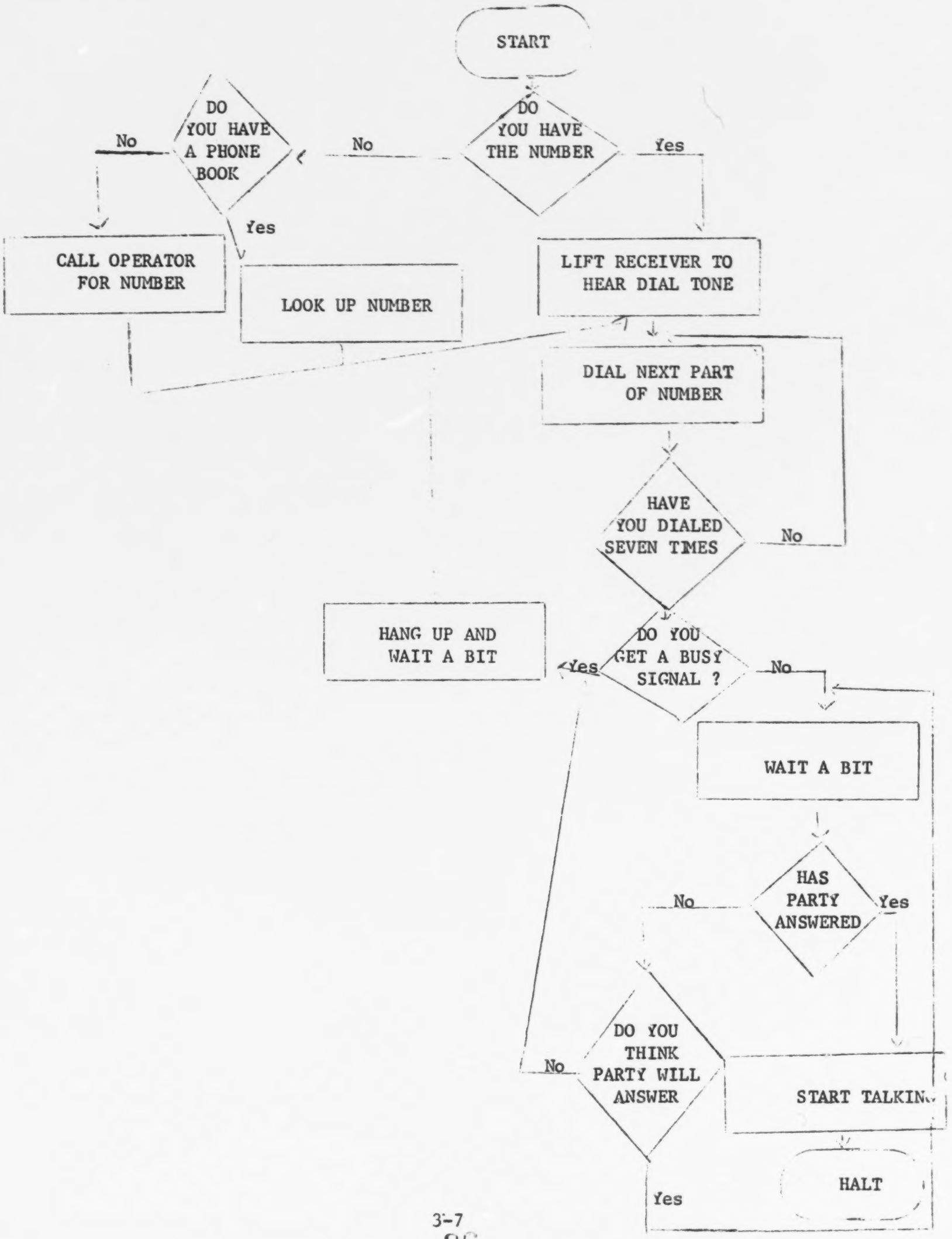


Flow Chart - Crossing An Intersection From A to B With No Automobile Traffic

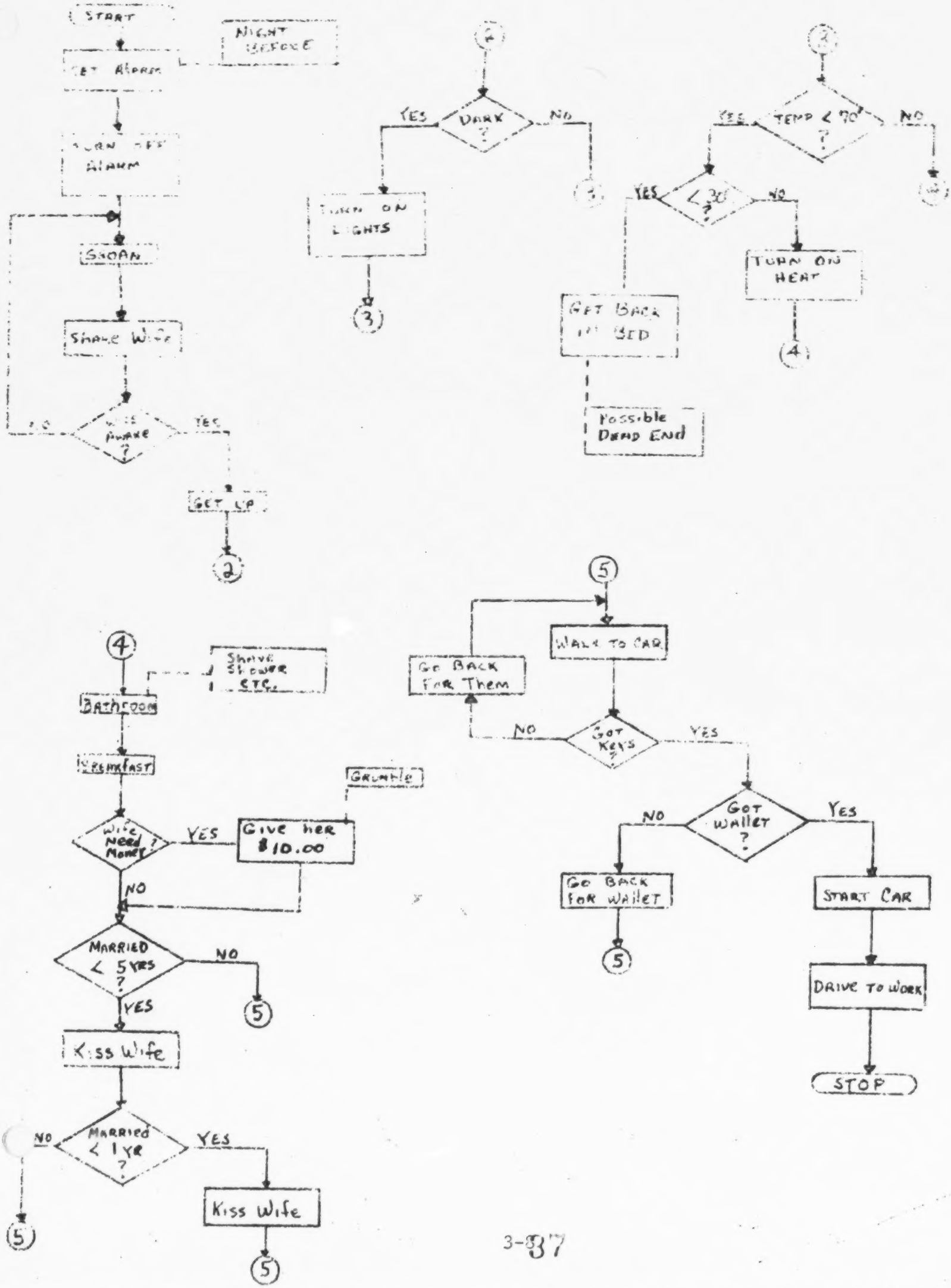


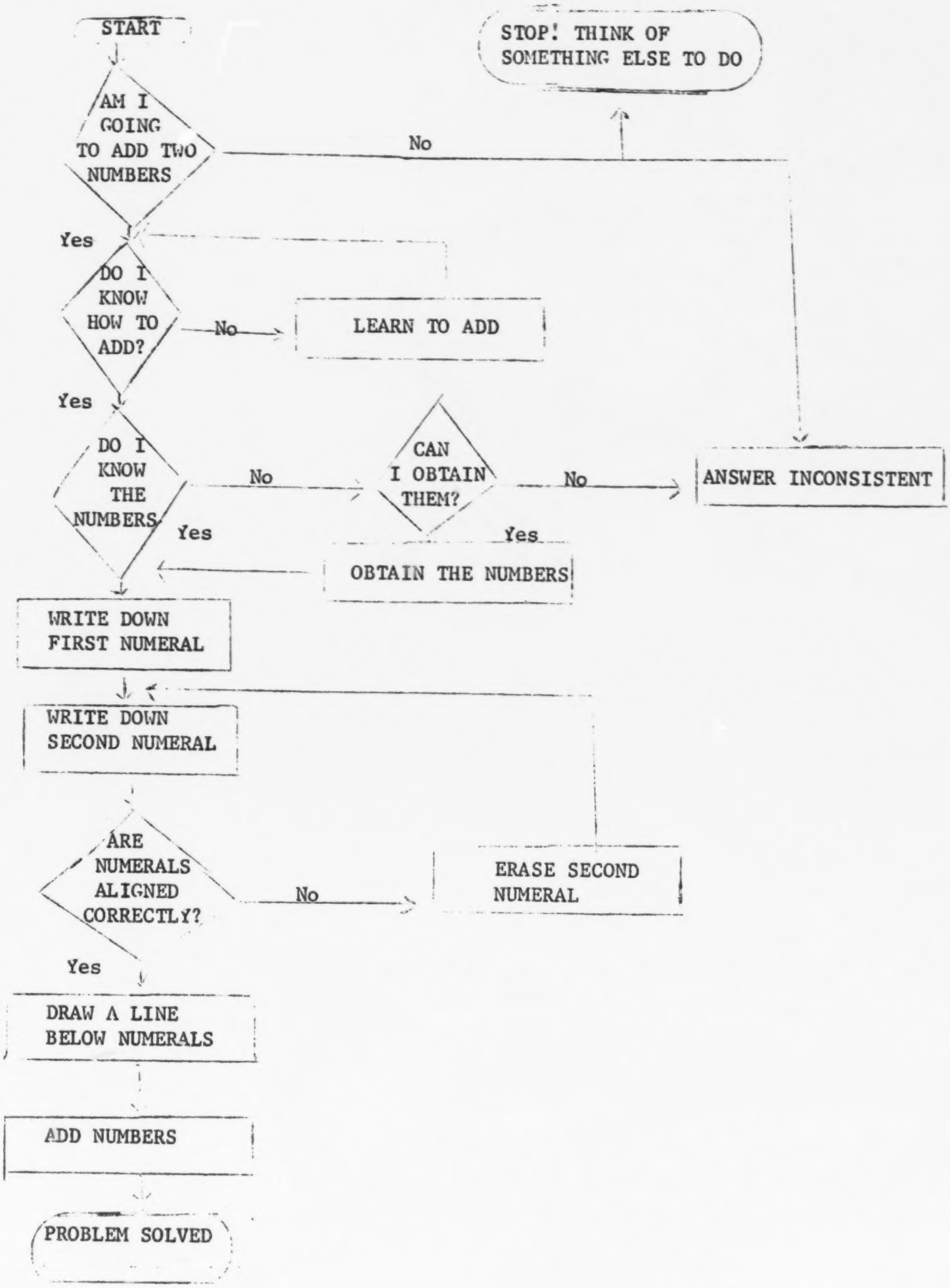


Flow Chart - Calling A Party On The Telephone

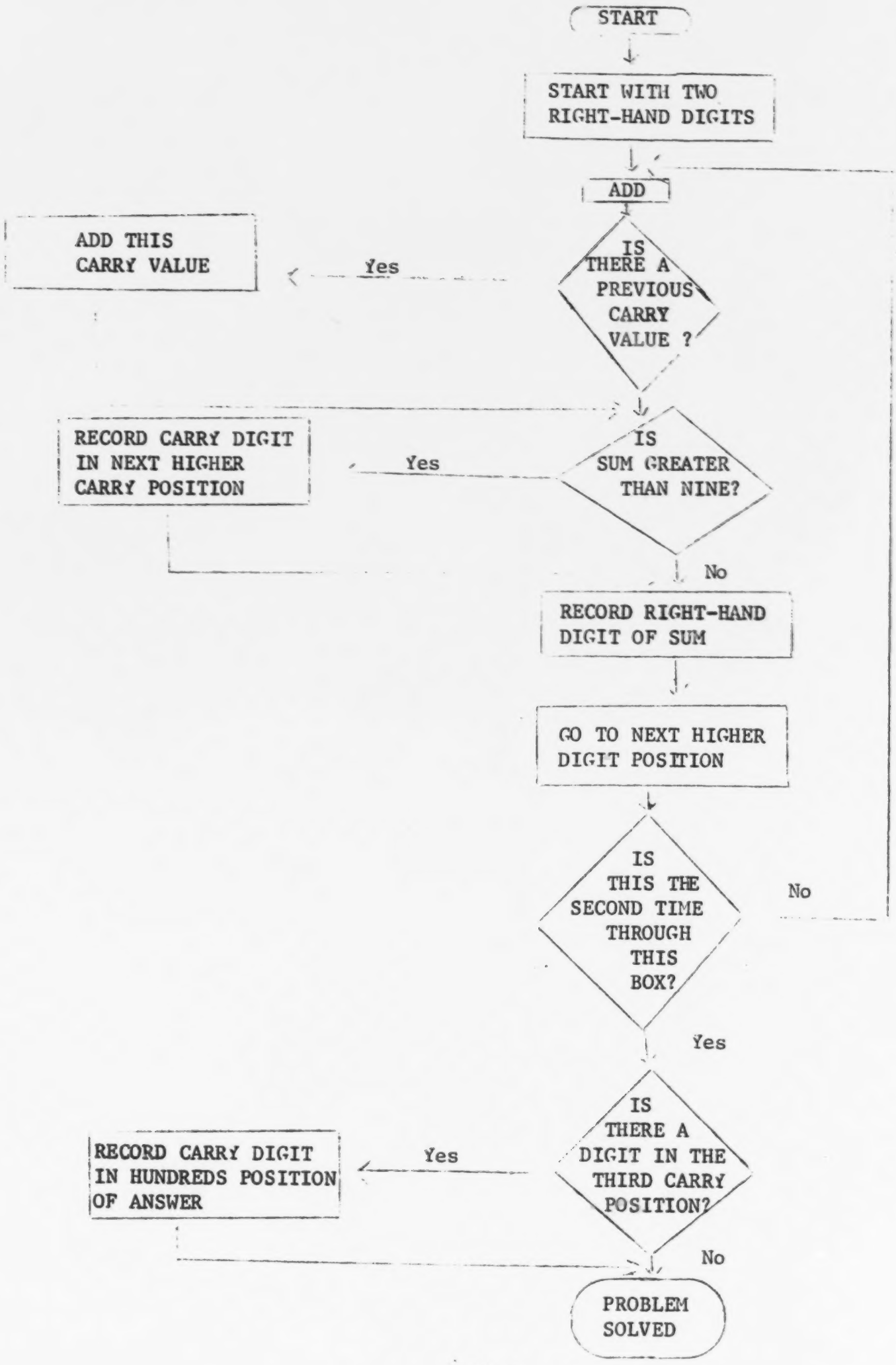


How To Get To Work




START
AM I GOING TO ADD TWO NUMBERS
No
STOP! THINK OF SOMETHING ELSE TO DO
Yes
DO I KNOW HOW TO ADD?
No
LEARN TO ADD
Yes
DO I KNOW THE NUMBERS
No
CAN I OBTAIN THEM?
No
ANSWER INCONSISTENT
Yes
Yes
OBTAIN THE NUMBERS
WRITE DOWN FIRST NUMERAL
WRITE DOWN SECOND NUMERAL
ARE NUMERALS ALIGNED CORRECTLY?
No
ERASE SECOND NUMERAL
Yes
DRAW A LINE BELOW NUMERALS
ADD NUMBERS
PROBLEM SOLVED

Flow Chart - Adding Two Two-Digit Numbers with Provision for Carry.



PROBLEMS:

1. Modify the Flow chart for the telephone call to include the discovery, when the party answers, of having dialed the wrong number.

2. Modify the Flow chart for the phone call to include the discovery, after dialing any digit of the number, of having dialed the wrong number.

3. Construct a Flow chart showing the process followed when a teacher takes attendance by calling the roll. Take into account the possibility of absentees.

4. Extend exercise number three to include the possibility that a student may enter the room unnoticed by the teacher after the roll-call has begun.

5. Construct a Flow chart showing the process of your getting ready for school in the morning.

6. Construct a Flow chart preparing a table of squares for numbers less than 10,000 by adding successive odd numbers.

7. Modify the Flow chart of the crossing of an intersection to include the possibility that there is a policeman who normally allows the traffic to flow according to the lights, but who may choose to direct it himself from time to time.

PROBLEMS TO FLOWCHART

1. Find the largest of three numbers, A, B and C.

2. Add two numbers. If sum is even, put 0 in the Accumulator. If add, put 1 in Accumulator.

3. Search locations 20_8 - $30_8$for all odd numbers in the range of 51_8 - 57_8 inclusive. Halt with the number of valves in this range in the Accumulator.

4. Solve AX + B > C for X.

*5. Multiply 4 by 5 by repeated additions.

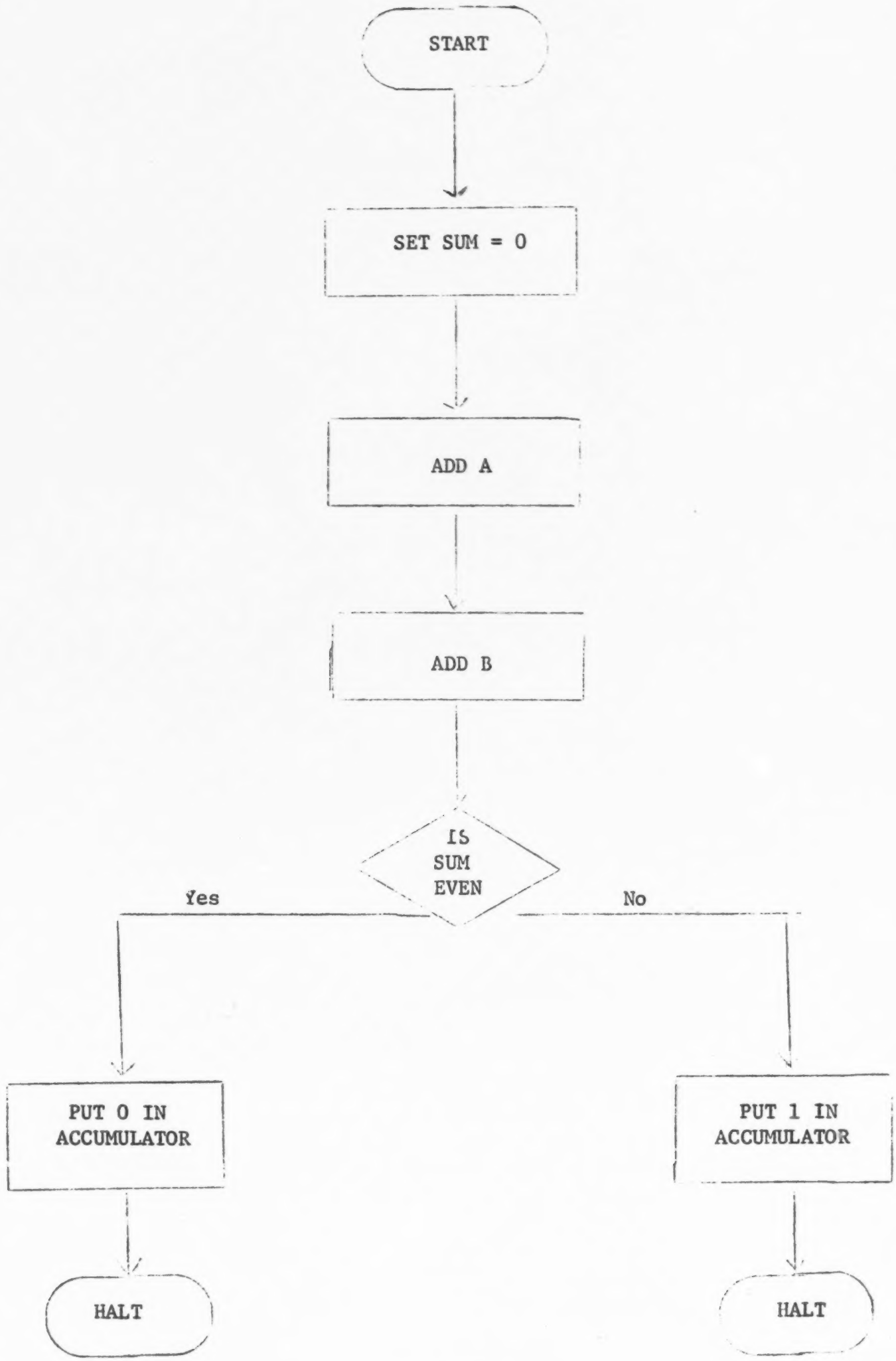
*6. Find the sum of the first ten natural numbers.

*7. Sum the contents of Core locations 30_8 - 50_8 . Store the sum in location 60_8 . Halt with sum in the Accumulator.

8. Find N! for a specified N.

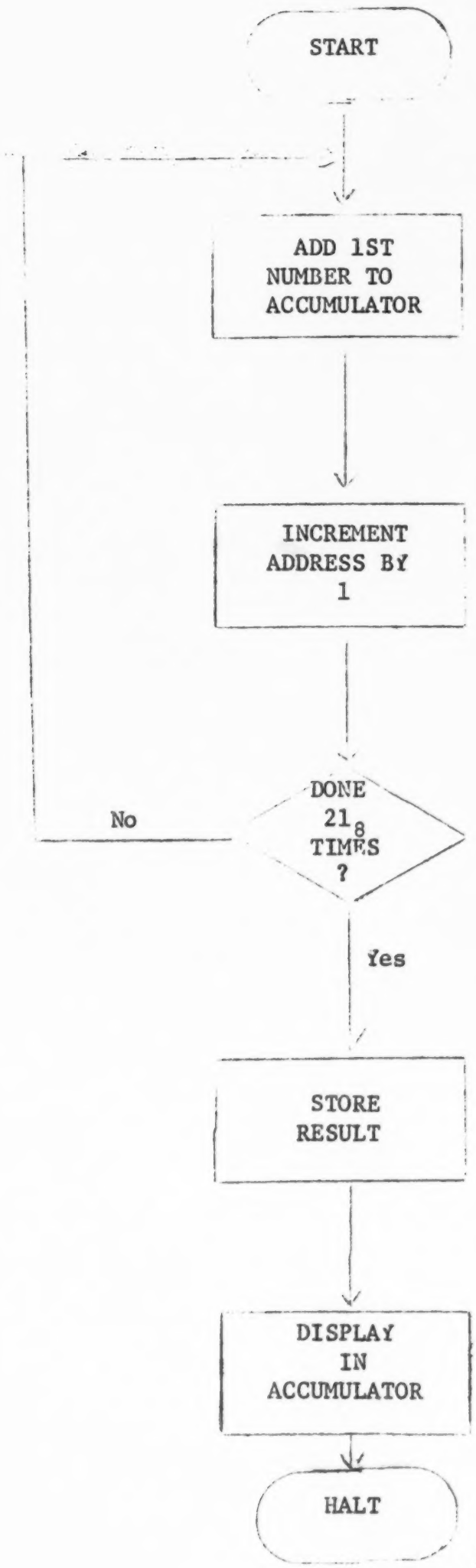
Problem #2

FLOW CHART



Problem #3

FLOW CHART



CHAPTER IV

COMPUTER EQUIPMENT

Ref: Introduction to Programming - Digital Equipment Corporation
Mavnard, Mass.

Chapter 1

COMPUTER EQUIPMENT

There are various ways by which you can communicate with the computer. The earliest method used and one that is still in use today is the punch card. They are capable of such tasks as college registrations cards, electricity bills, and payroll data. They are cheap and versatile. The cards have 80 vertical columns that can hold 80 characters of information (letter, numbers, or punctuation). One punch in a single column can represent any number from 0 to 9. Above the nine number rows are three zone rows: 0, 11, and 12. By making two punches in a single column - one in a zone row and the other in a number row, it is possible to represent any letter of the alphabet. Certain special characters can be represented by three punches in a single column such as a period is represented by a 12 punch, a 3 punch and an 8 punch. You must keep in mind that the symbolic language of the punch card must be translated into binary to be used by a computer.

The punched paper tape is cheap and easily stored in rolls. Typewriters and cash registers can be equipped to automatically punch paper tape as they are used in normal business. The resulting records can be easily fed into computers for further processing.

A third means by which you can communicate with the computer is the magnetic tape. The major advantages of the magnetic tape are:

a. it can hold huge quantities of data. A single 10 1/2 inch reel of tape can hold the contents of 250,000 punched cards, and
b. magnetic tape can read up to 120,000 characters per second from magnetic tape compared to only 1200 characters a second from punched cards.

Rolls of magnetic tape serve as the computer's external memory much the same as your text book serves you as external memory. The magnetic tape used in a computer system is similar to the kind used in a home recorder. It is a plastic ribbon with an iron oxide coating that can be magnetized. A tiny area of oxide is magnetized to indicate one in binary code; a blank area in the pattern stands for zero. The magnetic tape can be easily erased and re-used.

Some computer systems use magnetic discs for their external memory instead of the magnetic tapes. The discs, which look very much like a large phonograph record, can make information available to the computer more quickly than tape.

Magnetic ink allows for the reading of documents much the same way magnetic tape is read. The additional advantage of magnetic ink characters is that they can be read by people as well as by computers. Banks make wide use of magnetic ink characters, particularly on checks.

The optical scanner is a device that is currently under development. This machine would save much of the bother of translating information into special codes for computer input. An optical scanner has a set of electronic patterns in its memory and a photoelectric cell that scans the material to be

read converting the characters into electronic pulses. The scanner can "read" any character that matches the patterns stored in its memory.

The high speed printer is the most important medium for getting information out of the computer. A high speed printer prints information at a rate of up to 1200 lines a minute. For you to turn out 1200 lines, typing 60 words a minute on a typewriter, would take eight hours. The types of information that are turned out by this printer include accounting, forms, checks, and department store bills.

As part of their control unit, most computers have an electric teletypewriter. Through it the human operator can enter information directly into the computer. He can also receive through the teletypewriter short answers to problems solved by the computer. If there is a programming error, the computer may type out information that enables the human operator to make a correction.

Some computers have a section of lights that can instantly show what information is in control, memory and the arithmetic section. The display is in binary. If a light is on, it is a 1 and if the light is off, it is a zero.

The cathode ray tube is used when some kinds of scientific problems have answers that are in curves and other lines. While these answers can be printed, it is convenient to translate the information into an image that is flashed on a cathode ray tube. The human operator may learn what he needs simply by looking at the image or he may photograph the pattern for future use.

The equipment that makes up a computer system is called hardware. There are two categories of hardware: The central processor that is the actual computer, containing memory, control, and arithmetic units, and a variety of supporting equipment called peripherals.

All you need for a computer system is a central processor and at least one type of equipment to get information in and out of it. Generally speaking, the more kinds of peripheral equipment you have, the more kinds of jobs you can do.

The basic equipment that is needed for a computer system is to have a Central Processor and a console. The central processor is the actual computer. It contains a solid core memory, an arithmetic unit, and control circuits. There are no moving parts in the central processor and it operates at nearly the speed of light.

The console has all the controls that enable the human operator to run the computer system. It has the lights, switches, and a typewriter. It is through the typewriter that the operator sends information into the computor and receives answers out.

Some of the peripherals that expand the uses of the computer are the card reader, card punch, high-speed printer, magnetic tape units, and disc storage unit. The card reader reads the standard 80 column cards

column by column. As it reads, the information is automatically translated into the binary language of the computer.

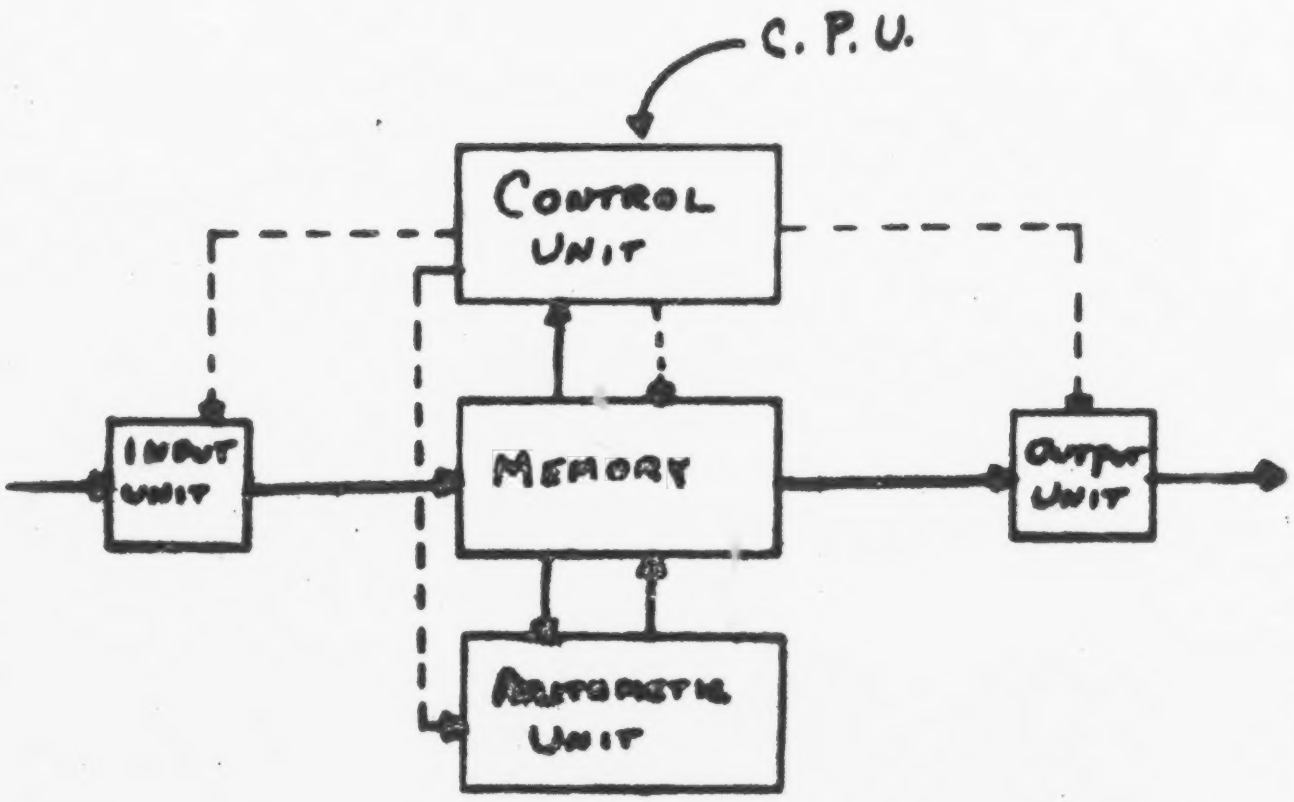
The card punch punches 80 column cards at 100 cards per minute. After punching each card, it reads the card to check for accuracy.

The high-speed printer is capable of turning out 1200 lines per minute. Unlike the typewriter, it prints an entire 136 character line at once rather than one character at a time.

The magnetic tape units, the computer's external memory, feeds information into the internal memory in the central processor as needed.

The disc storage unit is another external memory unit. It stores information on the magnetic discs for fast delivery to the central processor.

Basic Computer Block

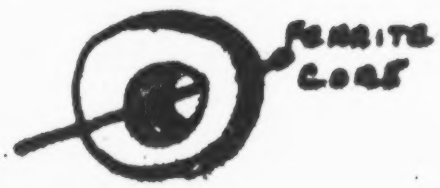
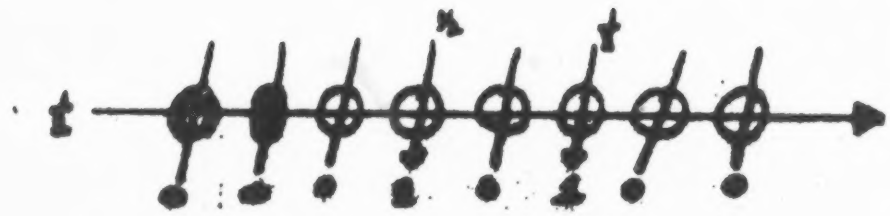


Input/Output

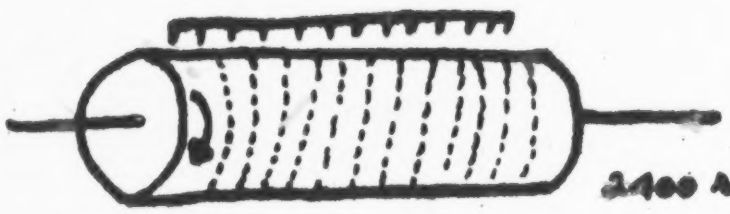
a) Teletype (10 characters/sec)
b) Magnetic Tape (40 million bits on a 10-inch reel)
c) Punched Cards
d) Disk
e) A-D Converters
f) Paper Tape

Card Readers (1000 cards/min)
Card Punches (300 cards/min)
High Speed Printer (1100 lines/min)
Optical Scanner (400-1600 documents/min)
Teletype (10 characters/sec)
Magnetic Tape (15,000 - 400,000 characters/sec)

MEMORY

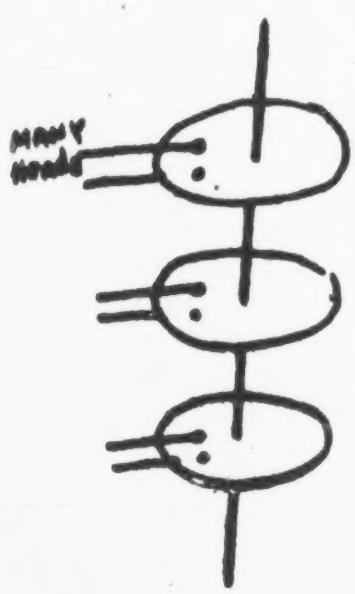


$10100_2 = 20_{10}$

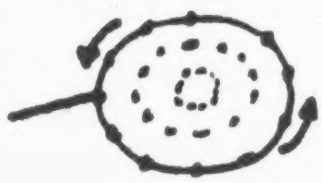


High Speed Drum (very fast)
(limited capacity)

2400 rev/min (2-3 million characters)



Disc Storage



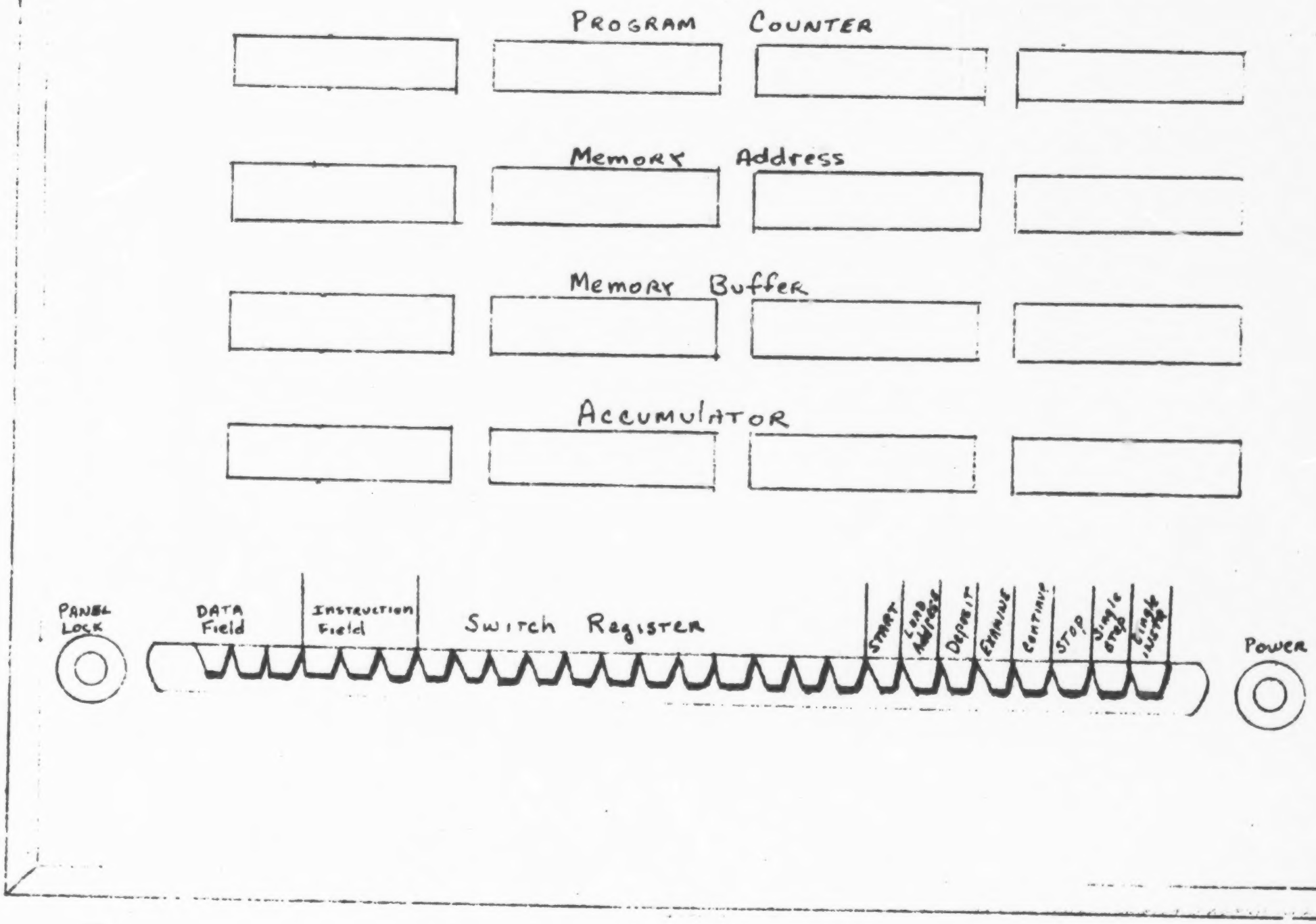
As head moves in it can pick up a concentric circle of information

A record is limited to 1 track on the disc (About 2000 characters)

(7.5 million characters)

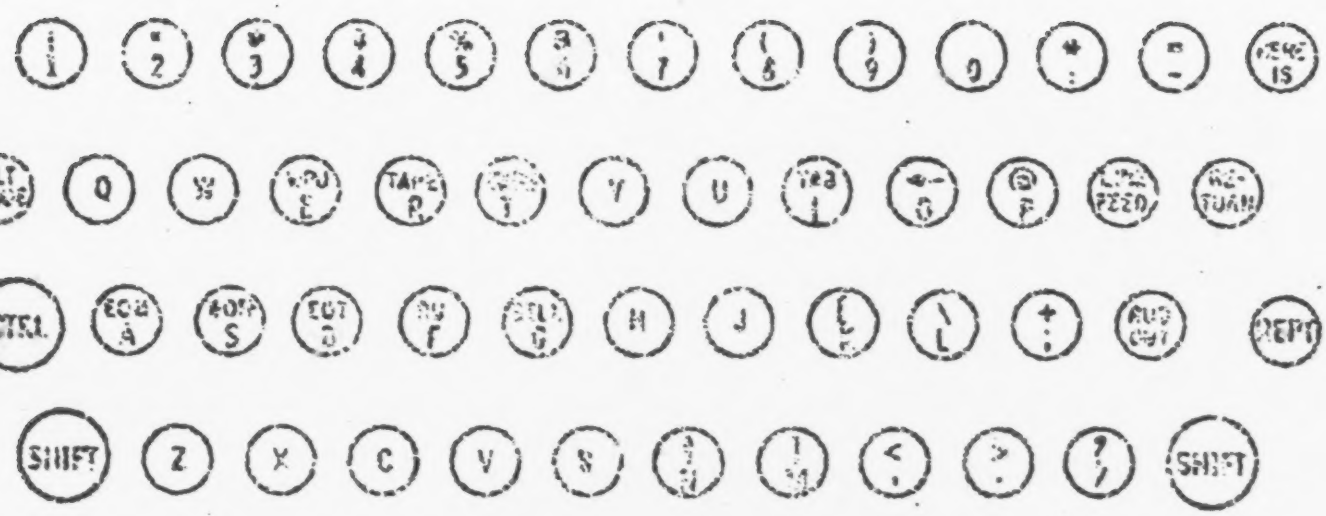
Magnetic Tape

(43 million characters)


PROGRAM COUNTER
Memory Address
Memory Buffer
ACCUMULATOR
PANEL LOCK
DATA Field
Instruction Field
Switch Register
START
LOAD ADDRESS
DEPOSIT
EXAMINE
CONTINUE
STOP
SINGLE STEP
SINGLE INSTR
Power

4-6

50



TELETYPE MODEL 33 KEYBOARD LAYOUT

CHAPTER 5

MACHINE LANGUAGE PROGRAMMING

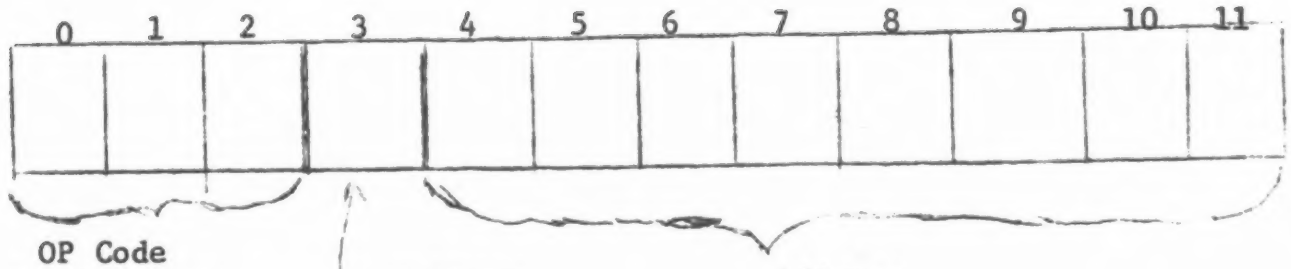
ASSEMBLY LANGUAGE PROGRAMMING

Ref. 1. Introduction to Programming - Digital Equipment Corp.
Maynard, Mass.

2. Programming Languages - Digital Equipment Corp.
Maynard, Mass.

Note to Teacher: When teaching this chapter it is useful to use an electric model of a computer word to illustrate the various instructions. An attempt should also be made to allow the students to run some programs through the console on the computer at Framingham State College or at Digital Equipment Corp. in Maynard. Both are receptive to this type of thing.

Instruction Word



00 000 000 → 0 → 0

.

.

11 111 111 → 377 → 255_{10}

^ 256_{10} possible core locations

Mnemonic	OP Code (Octal)	Meaning	
AND	0000	Logical and → AC	110 011 111 000
		Data	101 110 001 110
*TAD	1000	2's complement add	100 010 001 000
*ISZ	2000	Increment and skip next instruction if Zero	
*DCA	3000	Deposit accumulator and clear AC (store)	
JMS	4000	Jump to subroutine (program counter stored)	
*JMP	5000	Jump to a certain step	

CLA CLL 7300
HLT 7402

Explain

	Indirect
TAD	1
ISZ	1
JMP	1

TO RUN PROGRAM

Put starting address on kevs
Hit load address button
Push buttons down
Put in number for address
Hit deposit button

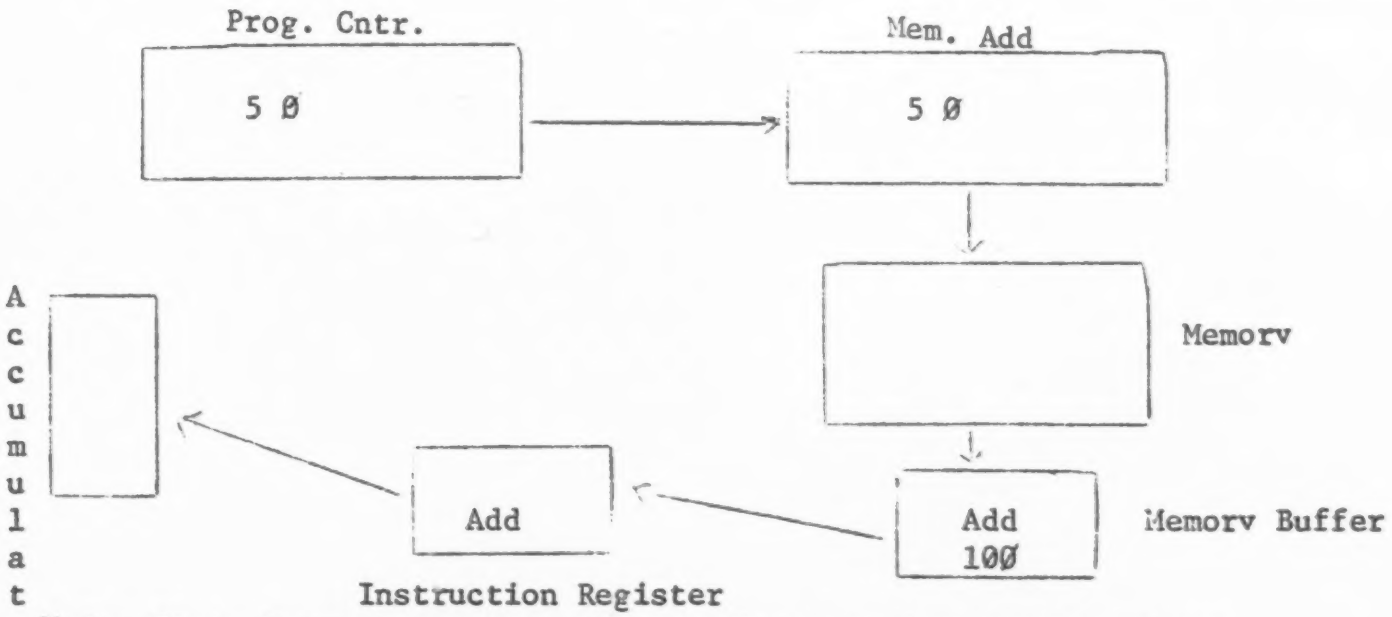
To Run
Put starting address
Hit load address
Hit start

5-1

Multiply by 2 → shift left 1 position

Divide by 2 → shift right 1 position

CORE	MEMORY		
0	1	2	3
		+5	- 10
4	5	6	7
	+ 30	- 76	
10	11	12	13



Major State Generator - goes into fetch state when program starts
the 50 in the PC goes into MA
the PC is increased by 1 → PC is now 51
Memory reads the contents of location 50 into MB
Instruction goes to instruction register. Now execute instruction in IR
Address portion (100) comes from MB into MA
Memory reads +2 into MB. The (+2) is shifted to AC, added to 0 and is left in AC

SAMPLE PROGRAM

Add 2 and 3

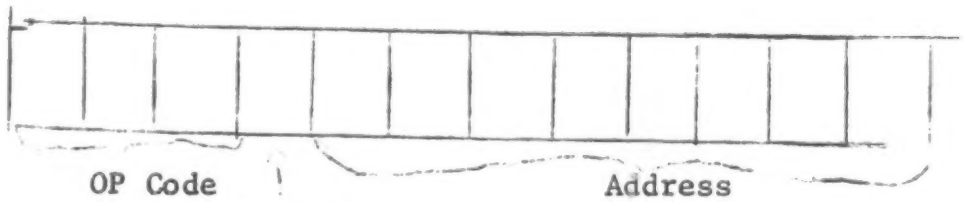
```
47/CLA CLL   ←  clears the accumulator and link

50/TAD  100  ←  means= the number in location 100 is added to the contents
                of the accumulator

51/TAD  105

52/DCA  107  ←  means = the number in the accumulator is put in location 107

53/HLT       ←  means program is finished

100/ +2

105/ +3      >      Storage location → you must put the numbers in which are to
                                       be added.  You don't have to put any
105/ 0                                 number in location 107 since the DCA
                                       instruction will replace whatever is there
                                       by the answer viz 5.
```

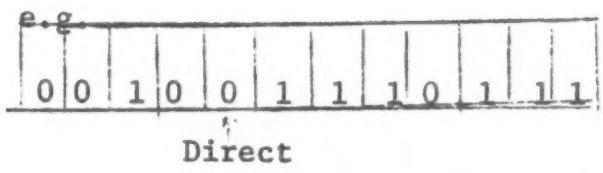
Note: You must insure that the accumulator is zero when you begin. This can be done by starting each program with the instruction CLA CLL

Note: Instructions come sequentially - hence if you tell the computer where the instructions begin, it considers everything an instruction until it sees a HALT

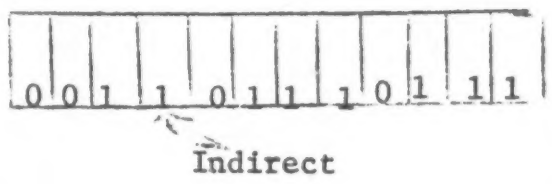
The above program would be entered through the console as follows:

BINARY	OCTAL
47/111 011 000 000	47/7300
50/001 001 000 001	50/1100
51/001 001 000 101	51/1105
52/011 001 000 111	52/3107
53/111 100 000 010	53/7402
100/000 000 000 010	100/0002
105/000 000 000 011	105/0003





TAD 167



TAD 1 167

166	167 2008	170
171	172	173
174	175	176
177	200 46	201

This savs add the contents of location 167 to the accumulator. Hence 2008 will be added to whatever number is in the accumulator.

This savs add the number which is found in the location contained in location 167. Hence the number to be added to the accumulator is the number which is found in location 200, namelv 46.

Problems: Program (with Flow chart) the following:

1. Add two numbers. If the sum is even, put 0 in the accumulator. If the sum is odd, put 1 in the accumulator.

2. Find 5 ! (5 factorial)

3. Write a program to compute the product of 3 and any other natural number.

4. Write a program to compute 6 x -7

5. Write a program to multiply 4 by 5.

6. Find the sum of the 1st ten numbers.

7. Sum the contents of core locations 30_8 through 50_8 and store the sum in location 60_8 . Halt with the sum in the accumulator. (indirect addressing)

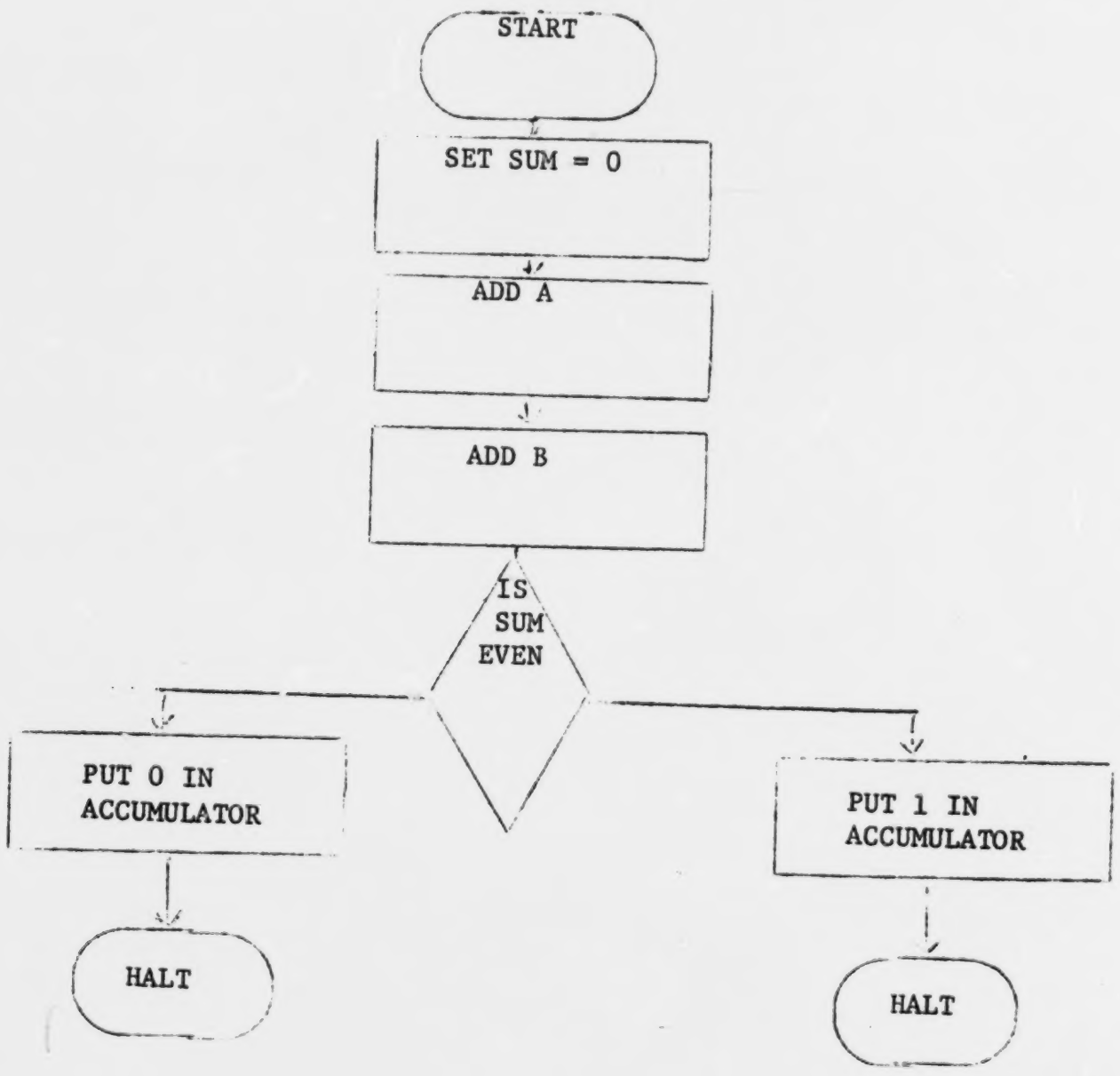
Problem L. Add two numbers. If sum is even, put 0 in the Accumulator. If odd, put 1 in the Accumulator.

```
200/ CLA  CLL
201/ TAD      100
202/ TAD      101
203/ AND      102
024/ HLT

100/  even, odd, odd
101/  even, even, odd

102/ +1
```

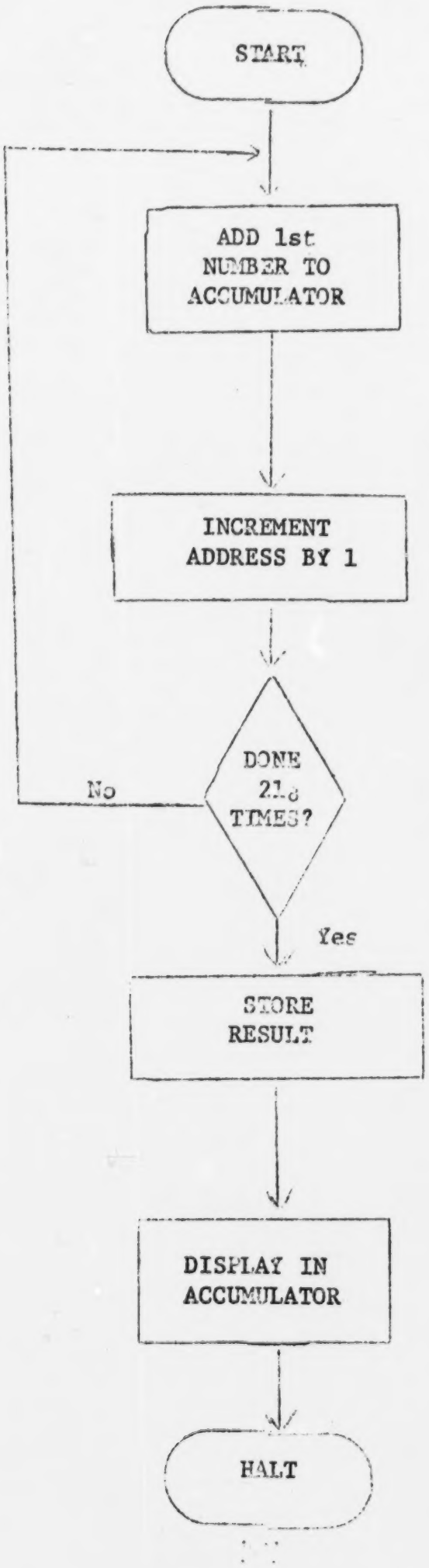
```
200/ 7300
201/ 1100
202/ 1101
203/ 0102
204/ 7402

100/
101/
102/ 0001
```

Run three times

Flow Chart (Problem #1)



FLOW CHART (Problem #7)



SOME OTHER INSTRUCTIONS*

CMA	7040	complement AC
CML	7020	complement link
RAR	7010	Rotate AC and link right one
RAL	7004	Rotate AC and link left one
RTR	7012	Rotate AC and link right two
RTL	7006	Rotate AC and link left two
IAC	7001	Increment AC by 1
SMA	7500	skip on negative AC
SZA	7440	skip on zero AC
SPA	7510	skip on positive AC
SNA	7450	skip on non zero AC
HLT	7402	halts the program
CIA	7041	complement and increment AC
LAS	7604	load AC with switch register
CLA IAC	7201	set AC = 1
CLA CMA	7240	set AC = -1
SZA CLA	7640	skip if AC = 0, then clear AC
SNA CLA	7650	skip if AC ≠ o, then clear AC
SMA CLA	7700	skip if AC < 0, then clear AC
SMA SZA	7540	skip if AC ≤ 0
SPA SNA	7550	skip if AC > 0
SPA CLA	7710	skip if AC ≥ 0, then clear AC

* A complete set of instructions will be found on the programmer's reference card and in the book INTRODUCTION TO PROGRAMMING (Appendix D)

PROBLEMS

8. Write a program which will put the larger of two numbers in the accumulator.

9. Write a program which multiplies a number by 2.

10. Write a program which divides a number by 2.

11. Write a program to accept two numbers from the switch registers. If they are equal halt with AC + 0 and if unequal halt with AC = -1.

12. Write a program to average two positive numbers entered through the console.

13. Write a program to interchange the contents of locations 300 and 301.

14. Add the first 8 odd numbers.

15. Add the numbers in locations 10 through 30.

16. Put the numbers 1 - 20_8 in consecutive locations starting at location 100_8

17. Search locations 100_8 to 150_8 to find the first number between 50 and 60. Put the number in the accumulator.

Problem #5 Multiply 4 by 5

```
200/CLA   CLL                    200/7300
201/TAD          101             201/1101
202/ISZ          102             202/2102
203/JMP          201             203/5201
204/HLT                          204/7402
101/4                            101/0004
102/-5                           102/7773   (2's complement -5)
```

Problem #6 Sum of 1st ten numbers

```
200/CLA   CLL                    200/7300
201/TAD          100             201/1100
202/ISZ          100             202/2100
203 ISZ          101             203/2101
204/JMP          201             204/5201
205/HLT                          205/7402
100/1                            100/0001
101/-10                          101/7766
```

Problem #7 Sum the contents of core locations 30_8 through 50_8 store the sum in location 60_8. Halt with sum in the Accumulator.

```
200/CLA  CLL          200/7300
201/TAD 1     250     201/1650
202/ISZ       250     202/2250
203/ISZ       251     203/2251
204/JMP       201     204/5201
205/DCA        60     205/3060
206/TAD        60     206/1060
207/HLT               207/7402
250/30₈               250/0030
251/-2₁₈              251/7757
```

Location	Number	Octal
30	1	1
31	3	3
32	5	5
33	7	7
34	9	11
35	11	13
36	13	15
37	15	17
40	17	21
41	19	23
42	21	25
43	23	27
44	25	31
45	27	33
46	29	35
47	31	37
50	33	41
	$289_{10} = 441_8$	

Ref: 1. A Guide to Fortran Programming - Daniel D. McCracken
John Wiley & Sons Inc. (New York)

2. Fortran Autotester - Robt. Smith and Dora Johnson
John Wiley & Sons Inc. (New York)

3. Fortran Programming Manual - Digital Equipment Corp.
Maynard, Mass.

4. The Bases of Fortran - Dr. R. E. Smith
Control Data Institute
Minneapolis, Minn.

5. Programming Languages - Digital Equipment Corp.
Ch. 15 Maynard, Mass.

CHAPTER 7
BASIC

References:

1. **Introduction to an Algorithmic Language (BASIC) - NCTM 1968 - $1.40**

2. **BASIC - An Introduction to Computer Programming Using the Basic Language**
   William F. Sharpe - The Free Press (New York)

3. **Problem - Solving With The Computer - Edwin R. Sage**
   Entelek Inc.
   42 Pleasant St.
   Newburyport, Mass. 01950

4. **Discovering Mathematics Through Computers - Dr. Viggo Hansen**
   Vol. 1 Rational Numbers (Part 1) **Olcott Forward Inc.**
   Vol. 2 Rational Numbers (Part 2) **234 N. Central Ave.**
   Vol. 3 Algebra **Hartsdale, N.Y. 10530**
   Vol. 4 Geometry
   Vol. 5 Advanced Algebra and Trigonometry
   Vol. 6 Probability and Statistics
   Price $4.00 each or $22.00 for the set of 6 (Add 50¢ for handling)

5. **Programming Languages Ch. 12 - Digital Equipment Corp.**
   Maynard, Mass.

BASIC

Symbols

+ addition
- subtraction
* multiplication
/ division
↑ exponentiation
i.e 2 ↑3 means 2^3

Order of Operation

1. Exponentiation
2. Multiplication
3. Division
4. Addition and Subtraction

Variables

A variable name is:

1. Any one of the 26 letters of the alphabet, or
2. Any one of the letters followed by a single digit.

e.g. A, Z, P, Q_4, R_6, X_2, etc. are acceptable variable names.

Program Statement

A program statement is one or more words in the BASIC language with the appropriate punctuation that tells the computer to do something. There can be only one statement per line in the program.
e.g. PRINT 3 + 4. is a statement which would cause the computer to print the number 7 when the program is run. (A program is a sequence of such statements.)

Note: Blank spaces in a program statement are not significant. Blanks may be freely used to increase legibility.

Line Numbers

Each statement in the program must be preceded by a line number. Any whole number from 1 to 99999 is allowed as a step number. The computer will process the program in ascending order of the line numbers (not in the order in which they are written)
Note: Line numbers need not be consecutive
e.g. A sample program might look like the following:

```
 14 PRINT  3 + 2
403 PRINT  5 ↑ 2
 84 PRINT  4 * 9
500 END
RUN
```

N.B. the computer would not do anything unless you type RUN. This tells the machine you are finished with your program and you want it executed.

In the above program the computer would type

```
 5      rather than      5
36                      25
25                      36  because
```

it executes the program in order of the line numbers.

IMPORTANT: When you number your steps in a program it is wise to use numbers like 10, 20, 30, etc. rather than 1, 2, 3, etc. since you can later insert written statements. The value of this will become obvious later.

Program Statements

Note: Each program statement must be preceded by a line number

I. PRINT

the PRINT command is followed by one of three things:

a. An arithmetic expression
e.g. PRINT 4 + 17 - this causes 21 to be printed.

b. an expression contained within quotation marks
e.g. PRINT "COST OF ITEM" - this causes COST OF ITEM to be printed.

c. a variable
e.g. PRINT X3 - this causes the value assigned to the variable X3 to be printed. For instance, if X3 = 58 then 58 would be printed.

Note: A combination of b and c is often employed
e.g. PRINT "THE AREA IS" Y - this causes the phrase THE AREA IS to be printed followed by the value of the variable Y.

Note: You may cause several things to be printed with one PRINT statement using commas.
e.g. PRINT "SOLN. SET =", X, Y

ASSIGNMENT

1. What numerical value will the computer give for each of the following:

   a. 18/6 * 3

   b. (18/6) * 3

   c. 18/ (6 * 3)

   d. 5 * 3 ↑ 2 + 4/2

   e. 5 * 3 ↑ (2 + 4/2)

   f. 5 * (3 ↑ 2 + 4) /2

   g. (5 * 3) ↑ 2 + 4/2

   h. 16 ↑ 1/2

   i. 16 ↑ (1/2)

   j. 8 - 5 - 3

   k. 8 * 3 - 2 + 1

   l. 18/3 * 6/2

   m. 4 ↑ 2.5

   n. 24/8/4

   o. 6 - 3/3

2. Write a program to evaluate and print the result of the following:

   a. $1 + \cfrac{1}{3 + \cfrac{1}{3 + \cfrac{1}{3}}}$

   b. $\cfrac{5^2 - 3^3}{7} + \cfrac{2}{5.3 - 6^2}$

   c. $\sqrt{21^2 - \cfrac{19}{(31.3)(1.87)}}$

3. Write a program which will label three columns - NUMBER, HALF, and DOUBLE, and will then compute and list under each column the number 7, its half (viz 3.5) and its double (viz 14)

PROGRAM STATEMENTS (Cont.)

II. READ

III. DATA

The DATA statement defines the numbers to be operated on in the program. There can be more than one number in each DATA line. They must be separated by a comma. You may have as many DATA lines as you need and they may appear anywhere in the program.

The numbers identified in the DATA statement have to be put into the program in order to be used. The READ command does this. For example, the statement

```
10 READ A
```

would cause the <u>first</u> number in the DATA list to be read into the computer and thereafter be identified as A.

IV END

The last statement in every program must be the word END. This is the signal to the computer that that is the end of the program. The END statement must have the largest line number of all the statements in the program.

SAMPLE PROGRAM

A sample program using the statements mentioned would be:

```
10 READ X, Y, Z
20 PRINT X, Y, Z
30 DATA  6, 3.2, 7
40 END
```

When executed, this program would cause the three numbers (6, 3.2 and 7) to be read out of the DATA statement and printed back on the teletype paper.

If you now typed RUN the program would be executed and the output would look like this:

```
RUN
 6             3.2           7

TIME:  0 secs  ←  this means it took less than 1 sec. to execute the program.
```

PROGRAM STATEMENTS (Cont.)

V. LET

The LET statement allows the user to define variables, e.g.

10 LET X = 5

causes the value 5 to be assigned to the variable X.

IMPORTANT: The equal sign in BASIC (and in most other computer languages) has a different meaning than in algebra. The equal sign instructs the computer to perform the computation on the right side of the equal sign and then store the result of this computation in a location identified by the variable name on the left side of the equal sign.
e.g. 40 LET A = X + 3 ↑ 2
This instructs the computer to add 9 to the value stored in X and then to store the sum in a location designated by A.

Remember that a variable name in a program refers to a location within the computer memory. Whenever that variable is called, the value stored in that location is entered into the computation.

Since the operations on the right side of the equal sign are performed first, a statement like

20 LET X = X + 1

has meaning. This causes 1 to be added to the value stored in X and this result to be stored back in X. That is, X now is its former value increased by 1.

ASSIGNMENT

1. Write a program which will read 5 numbers and print their average.

2. Write a program which will read 4 numbers and print their product.

3. Write a program which will read 4 numbers, printing the sum of the first two and the product of the last two.

4. Write a program which will read 4 numbers, printing the sum of the first two, the product of the last two, and the sum of the sum and product obtained.

5. Write a program which will evaluate $y = x^2 + 5x + 2$ when $x = 3$

6. Write a program which will evaluate $y = x^2 + 5x + 2$ when $x = 3$, -2, 4, 0 and 7

7. Write a program which will print the square and cube of -4, 6, 3, 0, 1 and -2

MORE ON SYMBOLS:

PARENTHESES - Operations within parentheses are done first. If there are parentheses within parentheses then the computer will perform the operations in the innermost parentheses first.

NOTE: The expression $\frac{x + y}{z}$ must be writen as (x + y)/Z.

The omission of parentheses would cause the expression $x + \frac{y}{z}$ to be read rather than the intended one.

EQUALITY AND INEQUALITY

= means equal
> means greater than
< means less than
<= means less than or equal to
>= means greater than or equal to
<> means not equal to

FUNCTIONS

The following functions may be included at any point in any legitimate arithmetic statement (e.g. a LET or PRINT statement). The results of the function can be further modified by multiplication, division, etc.

FUNCTION	RESULT
SQR (X)	The square root of X
ABS (X)	The absolute value of X
LOG (X)	The natural logarithm of X (lnx)
EXP (X)	E^X
SIN (X)	The sine of X (a number or an angle measured in radians)
COS (X)	The cosine of X (a number or an angle measured in radians)
TAN (X)	The tangent of X (a number or an angle measured in radians)
ATN (X)	The arctangent of X i.e. the angle (in radians) whose tangent is X
INT (X)	The integer portion of X. Any value to the right of the decimal point is dropped.
RND (X)	A random number between 0 and 1
SGN (X)	The sign of X { -1 if X < 0; 0 if X = 0; 1 if X > 0

Note: In the above functions, X is called the argument and must be included

within parentheses.

Note: The argument may be an arithmetic expression e.g. SQR (2 ↑ 3 + 6 * 12 + 1) will yield $\sqrt{81}$ which is 9.

Note: The argument of a function may also be a function e.g. INT (SQR (7)) will veild the number 2

Note: Functions may be used with other statements - e.g.

```
10 PRINT (-B + SQR (B ↑ 2 -4 * A * C )) / (2 * A)
                          or
20 LET X3 = SIN (2) + 7
```

Note: The argument of the random number function may be any variable - it's a "dummy" argument.

ASSIGNMENT

1. Write a program which will print the square root of 4, 5, 6, 9, 17 (use the SQR (X) function)

2. Write a program to compute the sine, cosine and tangent of 30^0 (approximate Π by 3.1415927)

3. Write a program to find the angle whose tangent is $\sqrt{3}$

4. Write a program to find the angle whose sine is 1/2

5. Write a program to find the arcsin of any number (you pick one)

6. Write a program to find the arccos of any number (you pick one)

7. Write a program to drop the decimal portion of the square roots of 10, 35 and 60.

8. Write a program to round off the square roots of 10, 35 and 60

9. Write a program to write 5 random numbers between 0 and 1

10. Write a program to write 5 random integers between 0 and 10

PROGRAM STATEMENTS (Cont.)

VI. FOR - NEXT

Aside: The real power of the computer becomes apparent when you can direct it to repeat the same operation or series of operations over and over again. This process is called looping or iteration. In BASIC, looping is accomplished by the use of two statements FOR and NEXT.

e.g.

```
10 FOR X = 1 TO 12
15 PRINT X, X ↑ 2
20 NEXT X
25 END
RUN
```

The above program will give the following output.

1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100
11	121
12	144

The program works this way: When you type RUN the computer reads the 1st number in the FOR statement, namely 1. It then prints 1 and the square of 1 It then is told to take the next value of X, namely 2, and then prints 2 and the square of 2. It continues this process until the last number in the FOR statement has been processed.

Note: The FOR statement need not begin at 1. For example:

```
10 FOR J = 5 to 8
20 PRINT 6 * J
30 NEXT J
40 END
RUN
```

The output would look like this:

```
30
36
42
48
```

Note: The FOR statement need not go up by increments of 1. If you wish a different increment you must specify it by the word STEP. For example:

```
10 FOR A = 3 to 5  STEP .5
20 PRINT A, A ↑ 2
30 NEXT A
40 END
RUN
```

The output would look like this:

```
 3         9
 3.5      12.25
 4        16
 4.5      20.25
 5        25
```

Important: There must be a NEXT statement for each FOR statement.

Note: There may be loops inside of loops. For example:

```
10 FOR B = 1 to 3
20 FOR C = 4 to 7
30 PRINT B, C
40 NEXT C
50 NEXT B
60 END
RUN
```

The output would look like this:

```
1          4
1          5
1          6
1          7
2          4
2          5
2          6
2          7
3          4
3          5
3          6
3          7
```

Note: The beginning and final values of the control variable do not have to be constants. They may be variables.
e.g.

```
10 READ A
20 READ B
30 FOR X = A to B
40 PRINT X/2
50 NEXT X
60 DATA 4, 7
70 END
RUN
```

This would cause the numbers 4, 5, 6, 7 to be divided by 2.

ASSIGNMENT

1. Write a program to print the first ten perfect squares.

2. Write a program to print the first twelve square roots.

3. Write a program to list the double of the integers from -5 to 2.

4. Write a program to list the values of Y in the expression $Y = X^2 - 5$ for the values of X from - 2 to 5 in steps of one half.

5. Write a program to list the products of the numbers 2 through 5 and the numbers 7 through 9. (Hint: a loop within a loop)

6. Write a program to add the first 15 numbers.

7. Write a program to add the first 8 perfect squares.

8. Write a program to print 20 random numbers.

9. Write a program to print the decimal equivalents for the unit fractions from $\frac{1}{2}$ to $\frac{1}{12}$ (i.e. $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$ etc.)

10. Write a program to print the sine of the angles from 0^0 to 360^0 in steps of 15^0. (Let $\Pi = 3.1415927$)

OUTPUT FORMAT

A word should be said about the output format at this time. Not only does the comma spearate variables in a PRINT statement, it also lines them up in columns.

There are five implied columns on the page. Up to five variables can be printed in one PRINT statement and they will be properly spaced across the page. Subsequent PRINT statements using variables separated by commas will also line up under these columns.

Labels may be printed over the appropriate columns by putting each in quotes and separating them with commas. There are a maximum of 15 PRINT positions for each column, so the heading labels must be 15 characters or less.

A comma at the end of a PRINT statement suppresses spacing after that line is printed so that several PRINT statements can print on the same line.

For example, the series of statements:

```
10 PRINT A,
   .
   .
   .
50 PRINT B,
   .
   .
   .
235 PRINT C
```

results in all three variables (A, B and C) being printed on the same line.

The word PRINT with nothing behind it can be used to insert additional spaces to make the output easier to read.

For example:

```
10 PRINT "NUMBER", "SQUARE", "CUBE"
20 PRINT
30 FOR N = 1 to 4
40 PRINT N, N ↑ 2, N ↑ 3
50 NEXT N
60 END
RUN
```

The output would look like this:

NUMBER	SQUARE	CUBE
1	1	1
2	4	8
3	9	27
4	16	64

The statement in line number 10 causes the labels to be printed. The PRINT in line number 20 causes a double spacing between the labels and the numerical output.

PROGRAM STATEMENTS (cont.)

VII. IF - THEN

The computer can make logical decisions within the program by comparing the value of two variables or two expressions. The statement to make these decisions is an IF - THEN. For example:

10 IF X > Y THEN 50 causes the program to jump to line 50 if X > Y. If X is not greater than Y the program continues with the next statement after 10.

Remark: The six relation symbols are

<	less than
< =	less than or equal to
=	equal
>	greater than
> =	greater than or equal to
<>	not equal

Note: You may use the IF - THEN to loop instead of the FOR - NEXT. To print the first ten numbers and their squares we could write:

```
10 LET N = 1
15 PRINT N, N ↑ 2
20 LET N = N + 1
25 IF N < 11 THEN 15
30 END
```

ASSIGNMENT

1. Write a program to put three numbers in ascending order.

2. Write a program to approximate the $\sqrt{2}$ without using the SQR function.

3. Write a program to list the divisors of 96.

4. Write a program to tell for what integers from 0 to 7 $2^n > N^2$
   (if less than or equal to, state which)

5. Write a program to tell when $2000, invested at 6%, compounded annually, will double.

6. Write a program to tell what the first integer N is which will make $2^n > 1{,}000{,}000$

7. Write a program to print 20 random numbers, writing the ones less than .5 in one column, the ones greater than .5 in another column, and the ones equal to .5 in a third column.

8. Write a program to print 20 random numbers, writing the ones between 0 and .3 inclusive in one column, the ones between .3 and .6 in a second column, and the ones between .6 and 1 inclusive in a third column.

PROGRAM STATEMENTS (Cont.)

VIII. GO TO

The statement GO TO (line number) causes the program to jump to the line number. For example:

40 GO TO 95 causes the program to jump to step 95. The example given under IF-THEN could have been written as follows:

```
10      LET   N = 1
15      PRINT N, N + 2
20      LET   N = N + 1
25      IF    N = 11 THEN 35
30      GO TO 15
35      END
```

IX. INPUT

In addition to reading numbers into the computer from DATA statements through the program statement READ, values may also be entered during the execution of the program by use of the statement INPUT followed by the desired variable name. For example, a program might contain the statement

90 INPUT X1

When the program reaches statement 90, a question mark will be printed by the teletype unit and the execution of the program will temporarily stop. You then type in the value of X1 you wish, hit the RETURN key on the teletype, and the program execution continues

Note: More than one variable may be input at one time simply by separating the variable names with a comma. For example:

15 INPUT A, B, C

asks for three numbers to be entered. Only one question mark will be printed, but three numbers, separated by commas, must be entered before the program execution continues.

ASSIGNMENT

1. Write a program (using GO TO) which will print a number and its cube until the cube exceeds 7000.

2. Write a program which will ask for 5 numbers and print their average.

3. Write a program which will ask for 4 numbers and put them in descending order.

4. Write a program to solve any quadratic equation ($AX^2 + BX + C = 0$) where the A, B, and C are asked for at the time of execution. If the roots are complex state this. If the user supplies A = 0 print out an error message to him. Call the roots X1 and X2 and have each root identified in the printout.

5. Write a program to have the computer select a random digit between 0 and 9 (without printing it), then ask the user to guess it, using such aids as "TOO HIGH", "TOO LOW", or "THAT'S RIGHT". Also, have the program terminate if the user takes more than 4 guesses.

6. Write a program that will ask the user to supply an amount of money to be banked, the rate of interest, the number of times compounded per year, and the number of years the money is left in the bank. Then print out the amount he would have at the end of this time. Use the formula $A = P\left(1 + \frac{r}{m}\right)^{mn}$ where P = original deposit, r = rate of interest, m = number of times compounded per year, and n = number of years.

Subscripted Variables

It is possible to assign a different variable name to each number to be used in a program. However, even for a relatively short list of numbers, this can produce a lengthy program.

There is a more powerful way to define variables - subscripting. If you had 20 numbers to work with, you can let X(1) be the first, X(2) be the second, X(3) be the third, etc.

Subscripted variables may be named by any letter, but they may not be followed by a number. That is A(7) is legitimate but A3(1) is not.

For example:

```
10 FOR I = 1 TO 6
15 INPUT  S(I)
20 NEXT  I

      .
      .
      .
```

This part of the program would allow you to put in 6 numbers, the first identified as S(1), the second as S(2), etc.

X DIM

The statement DIM means dimension and is required for all subscripted variables with more than 10 values. The statement is of the form:

```
10 DIM  S(25)
```

The number immediately following the variable name indicates the maximum number of subscripted values that will be used in the program. All of them do not have to be used.

Note: More than one variable may be mentioned in the same DIM statement. For example:

```
10 DIM  S(25), B(11), C(40)
```

Note: The DIM statement must precede the use of the subscripted variable.

ASSIGNMENT (use subscripted variables)

1. Write a program to print the first 20 Fabonacci numbers, labeling them U_1, U_2, U_3, etc.

   Note: $U_1 = 1$, $U_2 = 1$, $U_3 = 2$, -----, $U_n = U_{n-1} + U_{n-2}$, ----

2. Write a program to ask the user to supply 6 numbers and
   a. find their sum
   b. find their product
   c. find their average

3. Write a program to print a number, its square, its cube, its square root, and its cube root for the integers from 0 to 10.

4. Write a program to approximate one root of a cubic equation by Newton's tangent method. Use 5 approximations and have the user supply the A, B, C, D of $AX^3+BX^2+CX+D=0$ at the time of execution.

COMMANDS

LIST

Note: No line number required.
If you type LIST your program will be printed on the teletype paper. The line numbers will be printed sequentially.

Important: If you wish to change a line, one way it may be done is to type the same line number with the correct version you wish. The old line will be replaced by the new line. When you type LIST the new line will appear in the program and the old line will be deleted.

DEFINING FUNCTIONS: (DEF. FN-)

It is possible to define functions for use within any given program. A new function may be defined by the statement:

DEF FN - (variable) = expression
↖ a single letter

This means that up to 26 functions may be defined in a single program.
e.g. FNA, FNB, FNC, FND, etc.
For example if we wish to define a function which converts a percentage to its decimal equivalent, we could write

```
20 DEF FNP(X) = X/100
```

later when we wrote

```
50 PRINT FNP(37)
```

the computer would print out the number .37

An example of a program which might use the concept of a defined function would be the following:

```
10 FOR X=-3TO3 STEP .25
15 DEF FNF(X) = X↑2-3*X+7
20 PRINT X, FNF(X)
25 NEXT X
30 END
```

This program defines $F(X)=X^2-3X+7$ and finds the ordered pairs (X, F(X)) for all values of X from -3 TO 3 in steps of 1/4

SUBROUTINES (GOSUB - RETURN)

When a particular series of statements are to be performed more than one time in any given program, at different places within a program, the most efficient program uses a subroutine.

A subroutine, called by the statement GOSUB, is a series of statements that are performed and then control of the program returns to the statement immediately following the point where the subroutine was called.

For example, the statement

```
20 GOSUB 300
```

causes the program, when it reaches step 20, to go to the statements beginning at 300, continuing to the end of the subroutine as identified by the statement RETURN, and then returning to the first statement following the statement 20.

For example, the following program illustrates the use of a subroutine. It is a program to determine the GCD (greatest common divisor) of three integers using the Euclidean Algorithm. The first two numbers (A, B) are selected on lines 30 and 40 and their GCD is determined in the subroutine, lines 200 - 310. The GCD just found is called X in line 60, the third number (C) is called Y in line 70 and the subroutine is entered from line 80 to find the GCD of these two numbers. This number is, of course, the GCD of the three given numbers and is printed out with them in line 90.

```
10 PRINT  "A", "B", "C", "GCD"
20 READ    A, B, C
30 LET X = A
40 LET Y = B
50 GOSUB 200
60 LET X = G
70 LET Y = C
80 GOSUB 200
90 PRINT A, B, C, G
100 GO TO 20
110 DATA 60, 90, 120
120 DATA 38456, 64872, 98765
130 DATA 32, 384, 72
200 LET Q = INT (X/Y)
210 LET R + X- Q * Y
220 IF R + Q THEN 300
230 LET X = Y
240 LET Y = R
250 GO TO 200
300 LET G = Y
310 RETURN
320 END
```

MORE ON FORMAT

E Format

Numbers may be written in what is known as exponential format. For example, the number .0000123 can be written 1.23E-5 which means 1.23×10^{-5}

Other examples

10,000,000,000 can be written 1 E 10
.0000123 can be written .123E-4
.0000123 can be written 12.3E-6

This form is useful for extremely large or extremely small numbers. Any number to be used with BASIC may contain, at most, 9 digits. Hence, a number such as 10,000,000,000 must be written in E format.

Note: Small or large numbers are output by the teletype unit in exponential format. Any number less than .1 or greater than 100,000 is output in exponential form.

MATRICES

A matrix is a rectangular array of numbers. For example

$$\begin{pmatrix} 1 & 3 & -5 \\ 12 & 6 & 11 \\ 4 & 0 & 8 \\ 3 & 15 & 4 \end{pmatrix}$$

is a matrix with 4 rows and 3 columns. It is called a 4 x 3 matrix. A symbolic way of representing any 4 x 3 matrix would be

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{pmatrix}$$ where A_{ij} would represent the element in the i-th row and the j-th column.
e.g. A_{32} is the element in the 3rd row and 2nd column and in the first example is the number 15.

Basic provides a special set of twelve instructions for computations involving matrices. They are identified by the fact that each instruction must start with the word MAT. They are:

1.	MAT READ A, B, C	Read three matrices, their dimensions having been previously specified in a DIM statement. Data is stored in the matrix row by row.
2.	MAT C = ZER	Fill out matrix C with zeroes
3.	MAT C = CON	Fill out matrix C with ones

4.	MAT C = IDN	Set up matrix C as an identity matrix. (i.e. all ones down the diagonal, zeroes elsewhere.)
5.	MAT PRINT A, B, C	Print the three matrices, with A and C in the regular format, but B closely packed.
6.	MAT B = A	Set the matrix B equal to the Matrix A
7.	MAT C = A + B	Add the two matrices A and B and assign the value to C
8.	MAT C = A - B	Subtract the matrix B from the matrix A and assign the value to C.
9.	MAT C = A * B	Multiply the matrix A by the matrix B and assign the value to C
10.	MAT C = TRN (A)	Transpose the matrix A and assign the value to C
11.	MAT C = (K) * A	Multiply the matrix A, by K. K, which must be in parentheses, may be a formula. Assign the value to C.
12.	MAT C = INV (A)	Invert the matrix A and assign the value to C.

Dimensioning Rules

Each variable which is to be later used in a matrix operation must be defined in a DIM statement. For most of the MAT operations, this is simply to reserve enough space for the matrix.

The DIM statement may simply indicate what the maximum dimension is to be. Thus if we write:

```
DIM  M(20, 35)
```

then M may have up to 20 rows and 35 columns. This statement saves enough space for the matrix and hence the only care at this point is to assure that the dimensions declared are large enough to accommodate the matrix that must be taken.

The actual dimensions of a matrix are established either when it is set up by a DIM statement or when they are defined in one of four MAT statements:

```
MAT    READ
MAT    ZER
MAT    CON
MAT    IDN
```

e.g.

```
10 DIM M (20, 35)
50 MAT READ M (17,30) *
```

will read a 17 x 30 matrix since sufficient space has been reserved for it by statement 10

* some versions of BASIC have a zero row and a zero column so you would write MAT READ M (16, 29) to get a 17 x 30 matrix.

e.g. MAT M = CON (7,3) sets up a 7 X 3 matrix with 1 in every component.

The form of the four instructions is as follows:

```
MAT READ  C (M, N)
MAT C = ZER (M, N)
MAT C = CON (M, N)
MAT C = IDN (N, N)
```

where M and N are specifically stated or defined in the program by READ, INPUT, or LET

An example:

```
 10 DIM  A (2, 3)
 20 MAT  READ A
 30 MAT  PRINT A
         ------------
100 DATA 1, 2, 3, 4, 5, 6
200 END
```

Produces the following output:

```
 1                  2                  3
 4                  5                  6
```

If the DIM statement were changed to

```
10 DIM A (3, 2)
```

the output would appear as follows:

```
 1                  2
 3                  4
 5                  6
```

ASSIGNMENT

1. Write a program which asks you to input a 3 x 4 matrix and then prints it.

2. Write a program which asks you to input a 3 x 2 matrix and then prints it.

3. Write a program to evaluate a 2 x 2 determinant.

4. Write a program to print the solution of the system ax + by = c and dx + ey = d. (use determinants)

5. Write a program to evaluate a 3 x 3 determinant.

6. Write a program to evaluate a 4 x 4 determinant.

7. Write a program to add a 2 x 3 matrix to another 2 x 3 matrix.

8. Write a program to multiply a 2 x 3 matrix by a 4 x 2 matrix.

Note: Additional, advanced statements are to be found in the BASIC manual. Specialized printing rules, editing, and error diagnostics are there and the student is encouraged to read the explanations given and try to use each.

Of particular help will be the section on storing and recalling programs but students should not use storage except under unusual conditions.

CHAPTER 8

FOCAL

Ref:

1. Introduction to Programming - Digital Equipment Corp. Maynard, Mass.
2. Computers in the Classroom (A Resource Manual for Algebra) - Walter Koetke Digital Equipment Corp. Maynard, Mass.
3. Focal Programming Manual - Digital Equipment Corp. Maynard, Mass.
4. Programming Languages Ch II - Digital Equipment Corp. Maynard, Mass.

FOCAL

Order of Operations

1. Parentheses ( ), [ ], < >
2. Functions
3. Exponentiation ↑
4. Multiplication *
5. Division /
6. Addition and Subtraction +, -

Performs operations of same priority from left to right

Parentheses ( ), [ ], < >

Functions
later

Exponentiation
2 ↑ 3 means 2^3
2 ↑ n
limits on n
1) must be a non-negative integer (book says positive)
2) must be <2048 (book says <2046)

2 ↑ 3.262 → 8 Focal takes integral part

Problem: write the following the way it must be done using parentheses

$$\left(\frac{3\,(11.2 - 13.7^2)}{\frac{5.8^3 - 2.9}{13.87}} \right)^4$$

FORMAT

% → Floating point

± 0. bbbbbb E ± aaa

6 digits
no greater accuracy

won't be used unless needed
limits ± 619

+ 0. 999999 E + 619 biggest number
Ask what smallest is - answers will vary
Answer - 0.999999 E + 619

maximum is 19 (6 are significant, other 13 are zeroes)

% ab.cd

a = # of places to right of decimal place

a + b = total # of places

Problem: Write 123.456

% +0.123456 E + 03

% 9.03 + ␣ ␣ ␣ 123.456

% 5.02 + 123.46 ← rounded off

% 3 + 123

% 2 xx ← machine realizes it can't work in your format

Note: Some people may not know what $.32 \times 10^3$ is

$.32 \times 10^{3}$ is

etc

Review X^{-n}, X^{n}, $X^{\frac{m}{n}}$

You Are At The Teletype

(Assume Focal has been loaded in)

Machine Types

* Waiting for you

Re-turn ← You hit this if you want the machine to do something

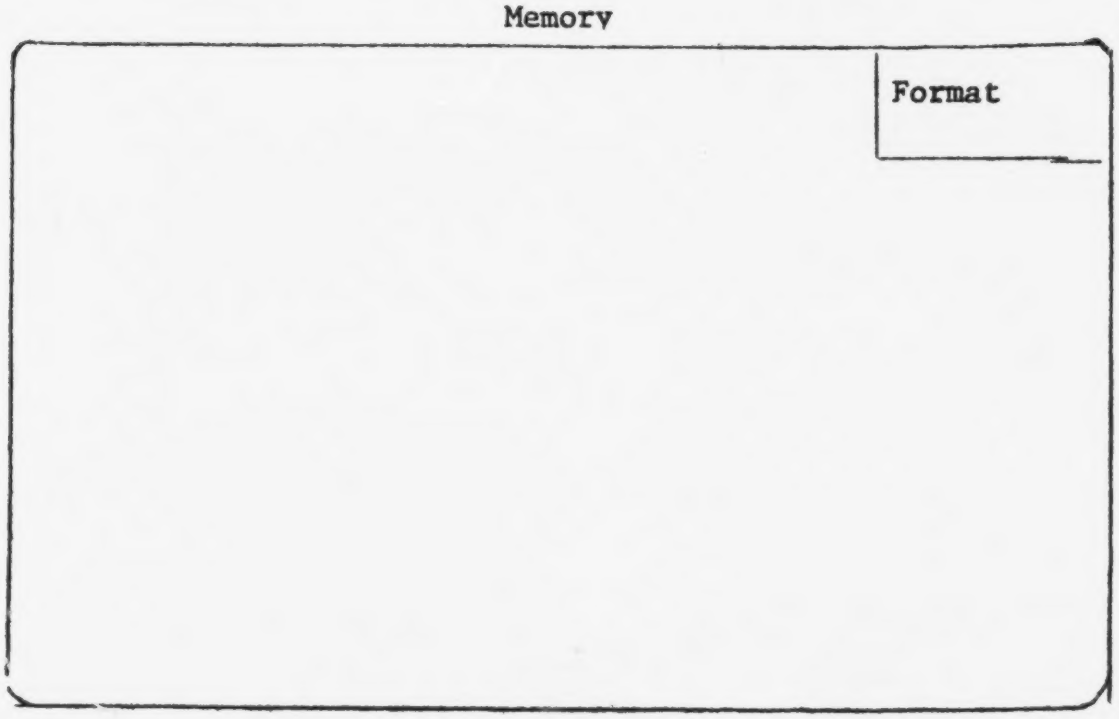
CTRL + C ← Hit these together to interrupt program. Returns program control to you.

* TYPE (Command)

```
      * TYPE  -  "JOHN DOE"                    RET
        JOHN DOE *                             RET
      * TYPE  ␣  "JOHN DOE",  "SUE SMITH"
        JOHN DOE  SUE SMITH *                  RET
      * TYPE  ␣  "JOHN DOE", !                  RET
        JOHN DOE                  Book doesn't use commas
      *
      * TYPE  ␣  "JOHN DOE",! " SUE SMITH",!   RET
        JOHN DOE
        SUE SMITH
       *
       *TYPE  ␣   "␣  ␣  ␣",  " JOHN DOE", "␣  ␣  ␣  ␣  ␣", "SUE SMITH",!   RET
                    JOHN DOE                      SUE SMITH

      *
      *
```

```
   * TYPE %, 3.15 + 23.6                    RET

=+0.267500E+02

   * TYPE % 5.02, 3.15 + 23.6               RET
=+ -  26.75

   * TYPE  5 ↑ 2                RET

+=  -  25.00
```

uses format from previous example



ERASE ALL → erases all input except last format

Must be careful in type statement if you omit format. It is possible that last format may not accommodate your answer.

* SET (Command)

SET ␣ variable name = Expression for which computer can get a numerical value

Any number of alpha-numeric characters
First Character must be a letter
First character cannot be F

However, machine only uses first two characters (i.e. All variables are identified in memory by a two character designation)

E.G.

```
SET     A = 3

SET     X1 = 4 ↑ 2

SET     MIKE = 3 + 6/2 ↑ 3

SET     PETE = A = X1 + MIKE      ← can define variables in terms of other variables

SET      A = A ↑ 2    ←   Can define a variable again (redefining)

SET      J = 4; SET K = 3;  SET Z = 2 ←can have more than one SET on a line
```

semi colons needed if more than one command is on a line.

ASSIGNMENT

1. What numerical value will the computer give for each of the following:

   a. 18/6*3

   b. (18/6)*3

   c. 18/(6*3)

   c. 5*3↑2+4/2

   e. 5*3↑(2=4/2)

   f. 5*(3↑2+4)/2

   g. (5*3)↑2+4/2

   h. 16↑1/2

   i. 16↑(1/2)

   j. 8-5-3

   k. 8*3-2+1

   l. 8↑-1

   m. 4↑2.5

   n. 24/8/4

   o. 6-3/3

   p. 18/3*6/2

2. How would FOCAL respond to the following commands if they were typed in the order given?

   a. TYPE %5.02, 48/24*2-2*24/48

   b. TYPE (1.2/(0.03*.4)) ↑1/2

   c. TYPE %, (4*10↑6/.2E-2)*3*10↑4

   d. TYPE (5↑2-2↑4)↑.5

   e. TYPE %3.01, 16.3+15.7

   f. TYPE "PETE", !!!,."MARY"

   g. TYPE 43.6/10/10

3. Write a FOCAL command to evaluate (and print the result of) the following

a. $1 + \cfrac{1}{3 + \cfrac{1}{3 + \cfrac{1}{3}}}$

b. $\cfrac{5^2-3^3}{7} + \cfrac{2}{5.3-6^2}$

4. How would FOCAL respond to the following commands if they were typed in the following order :

a.
```
SET A = 5
TYPE % 5.02, A
```

b.
```
SET BOOK = 3+2
TYPE BOOK
```

c.
```
SET Z2 = A+3
TYPE Z2 + BOOK
```

d.
```
SET PETE = 4
SET BOB = 5
TYPE %, PETE, BOB
```

e.
```
SET COUNT = -10
TYPE % 5.02, "COUNT = ", CO
```

f.
```
TYPE "A=", A, !, "BOOK", BOOK,!, "PETE = ", PETE
```

g.
```
TYPE  A, Z2
```

h.
```
TYPE A, "        ", Z2
```

i.
```
TYPE "A=", A, "           Z2", Z2
```

ERASE → erases all variable names

Indirect Commands

```
1.01            not allowed
1.02                    1, 2, 3, 4, 5, .....15
  .
  .
  .
15.99           group # (part #)  →  number to left of decimal
```

When line numbers are used, Focal stores entire line in memory.

*ASK (Command)
Allows user to input values for variables

```
     ASK X
     :   ←  machine waits for you to put in a value
ASK X, Y

    :3 ↑ :5
       hit a space or comma

ASK "X=", X, "Y=", Y
    X=:3   Y=:5

ASK  ?  X ␣ Y ?
        X:
        Y:  <—— Not in manual
                enclose requested variables, separated by spaced, between ?
                and you get them printed and on separate lines.
```

Problem: Ask for two number, get their sum and product.

Fixing Mistakes

```
1.1  ASK R
1.2 SET PI = 3.145159
                  ↑
                 Rub
                 Out   Erases 5
1.3 SET AREA - ␣ PI * \\\\= PI * R ↑ 2
               ↑
          Put in
          minus      Rub
          instead of = Out  5 times
```

```
1.4 ET CIRCUM = 2 * PI * R

    forgot S - Retype whole line

1.4 SET CIRCUM = 2* PI * R

1.5 QUIT

        forgot to get print out of answer - put a step number where you
        want the step to be

1.45  TYPE %, "AREA", AR, ! "CIRCUMFERENCE", CI, !!!
```

COMMAND

* WRITE → will type out the corrected program

* GO → will cause the program to be executed

* QUIT → program halts and user regains control

* ERASE ALL → erases all of memory (step and variables)

* ERASE → erases all variables

* ERASE 2 → erases all of part 2

* ERASE 1.7 → erases the step 1.7

FUNCTIONS

FSQT( ) → computes the square root of expression in parentheses
no space

FABS ( ) → gives the absolute value of expression in parentheses

FITR (N) → gives the greatest integer contained in parentheses
e.g. FITR (2.516) = 2 N<2048
FITR (0) = 0
FITR (-3.16) = -4

FEXP ( ) → computes e to the power within parentheses
e.g. FEXP (1) = 2.718281 e + 2.718281

FLOG ( ) → computes natural log of # within parentheses
Review rules of logarithms
$\log AB = \log A + \log B$
$\log \frac{A}{B} = \log A - \log B$
$\log A^n = N \log A$

FUNCTIONS (cont)

e.g. FLOG (1.98765) = .686953 → e .686953 = 1.98765
FEXP (.686953) = 1.98765 → e .686953 = 1.98765
Hence FEXP (FLOG (A)) = A

FSIN (N) → computes the sine of an angle (N must be in radians)

Conversion

$D \cdot \frac{\Pi}{180}$ = angle in radians

$R \cdot \frac{180}{\Pi}$ = angle in degrees

FCOS (N) → computes the cosine of an angle (in radians)

FATN (N) → computes the angle in radians whose tangent is the value N

e.g.

FATN (1) = $\frac{\Pi}{4}$

FATN ($\sqrt{3}$) = $\frac{\Pi}{3}$

ASSIGNMENT

1. Write a Focal command which will evaluate and type each of the following expressions:
   a. SIN 60^0
   b. COS^2 3.1
   c. ARCTAN (.31) ← express answer in degrees
   d. $3\sqrt{}\overline{3\sqrt{2.9}}$

2. Write a program which asks for two positive integers, N and D, and then type the quotient of the division of N by D, then type the remainder of the division.

3. Write a program which asks for two numbers, A and B, and then writes their sum and product.

4. Write a program which asks for two sides of a right triangle and then type the two possible lengths of the third side.

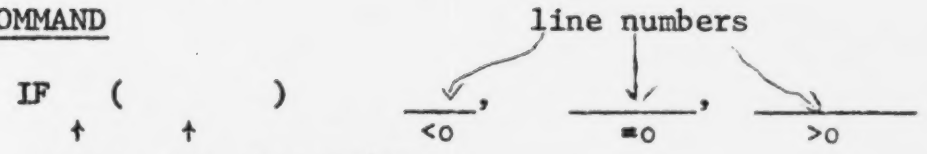
5. Write a program which asks for five integers and then types their average.

6. Compound interest : If $A = P\left(1 + \frac{R}{M}\right)^{mn}$ where

   A = amount in bank
   P = original principal deposited
   R = rate of interest
   M = number of times compounded per year
   N = number of years money is left.

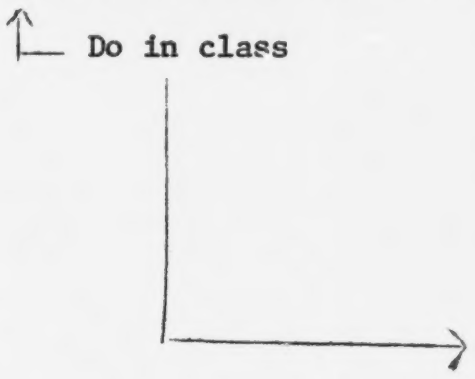
   Write a program which asks for P, R, M, and N and then types amount in your bank account.

* COMMAND



computer evaluates expression and compares it to zero

Problem: ask for two numbers A and B, divide A by B, tell if quotient is positive, negative or zero.




```
ERASE ALL
1.1 ASK A, B, !
1.2  IF (B) 1.5, 1.3, 1.5
1.3  TYPE "UNDEFINED"
1.4  QUIT
1.5  IF (A/B) 1.6, 1.7, 1.8
1.6  TYPE "NEGATIVE"
1.65  QUIT
1.7  TYPE "ZERO"
1.75  QUIT
1.8  TYPE "POSITIVE"
1.85 QUIT
GO
```

Now do same problem putting many commands on one line separated by semicolos.

```
1.1  ASK A, B, !; IF (B) 1.3, 1.2, 1.3
1.2 TYPE "UNDEFINED"; QUIT
1.3 IF (A/B)  1.4, 1.5, 1.6
1.4  TYPE "NEGATIVE"; QUIT
1.5  TYPE "ZERO"; QUIT
1.6  TYPE "POSITIVE"; QUIT
```

Problem: Ask for A and B, type the larger

ASSIGNMENT

1. Write a program which asks for two numbers and then tells whether the sum or the product is the larger and if they are equal it states this also.

2. Write a program which asks for an integer and tells if it is even or odd.

3. Write a program that asks for three numbers and prints the largest.

4. Write a program which asks for A, B and C in the quadratic equation $AX^2 + BX + C = 0$ and solves the equation.

5. Write a program which asks for an integer and tells if it is divisible by 7.

6. Write a program which asks for two natural numbers N and D and then tells if N is divisible by D.

* COMMAND

GOTO ⌴ ⌴ program will transfer control to a specific line number and continue from that point

line number

Problem: Ask for a number and tell if it is in position, negative or zero; even or odd; integer or not an integer.

```
1.1  ASK ? N ?, !
1.2  IF (N)  1.3, 1.5, 1.4
1.3  TYPE "NEGATIVE",!; GOTO ↵ 1.6
1.4  TYPE "POSITIVE",!; GOTO ↵ 1.6
1.5  TYPE PZERO, EVEN, INTEGRAL",!;OUIT
1.6  IF (FITR(N)-N) 1.8, 1.7 ← comment not needed
1.7  TYPE "INTEGRAL",!; GOTO ↵ 1.9
1.8  TYPE "NON-INTEGRAL",!; OUIT
1.9  IF (FITR (N/2) - N/2) 1.95
1.91  TYPE "EVEN", !!!; OUIT
1.95  TYPE "ODD", !!!; OUIT
```

Problem: Write a program to ask for an integer and tell if it is a prime.

ASSIGNMENT

1. Write a program which will ask for N numbers and take their average.

2. Write a program which will ask for N numbers and tell the smallest.

3. Write a program which will ask for N numbers and then types the number and its square in two columns headed "NUMBER" and "SQUARE"

*4. Write a program which will give the first N Fibonacci numbers.

*5. Write a program which will find the G.C.D. of N natural numbers.

*6. Write a program which will ask for N numbers and will then compute and type the mean and standard deviation of the numbers.

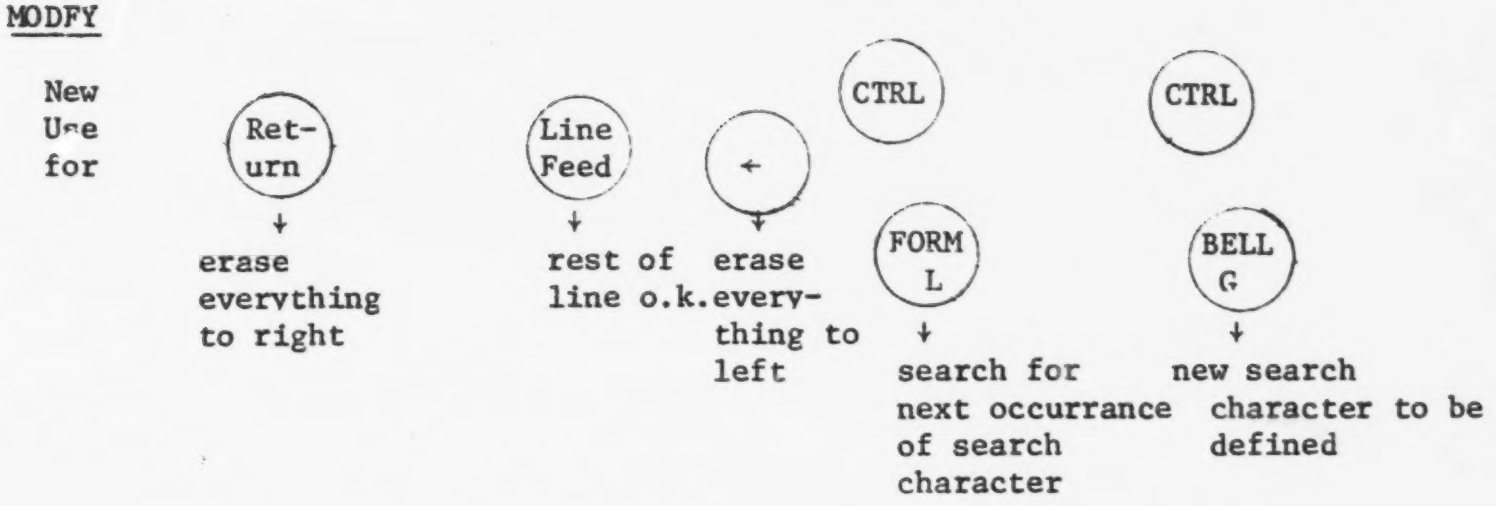
$$\text{MEAN} = \frac{\sum_{i=1}^{N} ai}{N} \quad \leftarrow \text{(the average)}$$

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^{N} (Ai - \text{Mean})^2}{N}}$$

*7. a. Write a program which will compute and type N factorial (where N is supplied at execution)

* b. Write a program which will compute and type a number and the factorial of that number from 1 to N (where N is supplied at execution)

MODFY



1.3 IF (A-2B) 1.4, 1.5, 1.6

* MODIFY 1.3 (Ret)

↑→ You define search character which will not be printed.

Modify erases variable table.

ASSIGNMENT

Correct (using Modify Command) each of the following.
After each correction say TYPE 1.1, TYPE 1.2, etc. to show corrected form.

1. You want 1.1 ␣ SET ␣ X= 2*Y; SET ␣ P=P+2; TYPE ␣ X, Y, !
   You wrote 1.1 ␣ SET ␣ X = 2Y; SET ␣ P=P+2; TYPE ␣ X, Y, !

2. You want 1.2 ␣ IF ␣(A-2*B) 2.2, 2.3, 2.4
   You wrote 1.2 ␣ IF ␣ (A-2*B) 2.2 2.3, 2.4

3. You want 1.3 ␣ SET ␣ X = A+B; SET ␣ M = A*B; TYPE ␣ X, M, !
   You wrote 1.3 ␣ SET ␣ X=A+B; SET ␣ M=A**B; TYPE ␣ X, M, !

4. You want 1.4 ␣ IF ␣ (A-2*B) 1.8, 1.8, 1.9
   You wrote 1.4 ␣ IF ␣ (A-2*(1.8, 1.8, 1.9

5. You want 1.5 ␣ SET ␣ A=B+1; SET ␣ C+A+2; IF ␣ (A-C) 1.2, 1.3, 1.4
   You wrote 1.5 SET ␣ A=B+1; SET ␣ C=A+2; CF ␣ (A-C) 1.2, 1.3, 1.4

6. You want 1.6 ␣ SET ␣ M=2*B; IF ␣ (M-N) 1.3, 1.4, 1.5
   You wrote 1.6 ␣ SET ␣ N ␣ 10; SET ␣ M = 2*B; IF ␣ (M-N) 1.3, 1.4,

7. You want 1.7 ␣ SET ␣ P=P+2; SET ␣ O=O-1; TYPE ␣ P, O, !
   You wrote 1.7 ␣ SET ␣ P=P+2; SET ␣ O=O-1; TYPE ␣ P, O; GOTO ␣ 1.4

8. You want 1.8 ␣ TYPE ␣ % 5.02, A, B, !
   You wrote 1.8 ␣ SET A=3; TYPE ␣ A, B, !

C. COMMENT

Line # → is not printed each time program runs

SUBSCRIPTS (single)

A_2	A [2]
MAX_y	MAX [Y]
X_{i+j}	X [I+J]

Variable Name [↑]
expression, number, variable

[is (vt / k) shift

] is (M) shift

Problem: write a program which will
1. ask for N
2. ask for N numbers
3. tell which is the largest

```
1.1    ASK N,!!; SET I=1
1.2    ASK A[I]; SET I=I+1; IF (N-I) 1.3, 1.2, 1.2
1.3    SET LARGE = A[1]; SET J=2
1.4    IF (A[J]-LARGE) 1.5, 1.5; SET LARGE=A[J]
1.5    SET J=J+1; IF (N-J) 1.6, 1.4, 1.4
1.6    TYPE "LARGEST IS", LARGE, !!; QUIT
```

FOR and DO

```
1.05  TYPE     "NUMBER", "                    SOUARE", !
1.1   FOR N=1, 10; DO 2
1.2   QUIT
2.1  TYPE %3, N,"              ", N↑2,!
*GO
```

NUMBER	SOUARE
=1	=1
=2	=4
=3	=9
=4	=16
=5	=25
.	.
.	.
.	.
=10	=100

Problems:

1. Ask for N numbers + show advantage of previous system.

2. Write a program to ask for N, then N terms A_1, A_2 , --- A_n then type the sum of the series
   SUM = $A_1 + 2A_2 + 3A_3 + 4A_4 + \text{---} + NA_n$

3. Sum the first N odd numbers

4. Get N numbers, type largest and smallest.

5. Sum of N numbers.

DO - transfers control to a line or a part

1.2 DO ␣ 3.5 → will do line 3.5 and will then go back to the command following the DO

2.4 DO ␣ 4 → will do all of part 4 and return to the command following DO command.

RETURN → will terminate the DO subroutine and control is transferred to the command following the DO

e.g.

```
1.1 SET A = 2; SET B = 3
1.2 DO 2
1.3 TYPE A*B,!
1.4 DO 3; DO 2.2
1.5 QUIT          ← only does this line

2.1  SET A=$
2.2  TYPE A+B,!
2.3  TYPE A-B,!
2.4 RETURN  ←  you're finished with DO statement
2.5  SET B = 6 ← never does this

3.1  SET A = 3
3.2  TYPE A/B,!
*GO
```

~~A = 2~~ B = 3

~~A = 4~~
~~A = 3~~

OUTPUT

= 7
= 1
= 12
= 1
= 6

GOTO 2.2 goes to 2.1 and continues from this point

DO 2.2 goes to 2.2, does it and returns to command following DO

```
1.1  DO 2
2.1  DO 3
2.2  TYPE ␣ ␣ "Here ␣"
3.1  DO 4
3.2  TYPE " ␣ I ␣ "
4.1  TYPE " ␣ AM ␣ "

* GO
```

Focal types Am I here Am I Here Am I Am
to make it type Am I Here we could

a. use DO 1 instead of GO or
b. have 1.2 QUIT

```
1.1  ~~DO 2~~  Am I Here
     ~~DO 3~~
     ~~DO 4~~
2.1  ~~DO 3~~  Am I Here
2.2  ~~DO 4~~
3.1  DO 4   Am
3.2         I
4.1         AM
```

Ex. 1

```
1.1 TYPE "A"
1.2 DO 2
1.3  TYPE "C"

2.1 TYPE "B"
2.2 GOTO 2.4
2.3 TYPE "D" ← never printed
2.4 TYPE "E"
2.5 TYPE "F"

3.1 TYPE "G"
3.2 TYPE "H"
```

ABEFCBEFG

Ex 2

```
1.1 TYPE "A"
1.2 DO 2
1.3 TYPE "C"

2.1 TYPE "B"
2.2 GOTO 3.1
2.3 TYPE "D"
2.4 TYPE "E" ← never printed
2.5 TYPE "F"

3.1 TYPE "G"
3.2 TYPE "H"
```

ABGCBGH

Note: If a GOTO (or IF) transfers to a line in the DO group (Ex 1) the remaining lines are executed before returning to command following the DO.

If a GOTO (or IF) transfers to a line outside the DO group (ex 2) that line is executed and control is returned to the Command following the DO.

FOR

```
FOR A = B, C, D;  command
        ↓
      first  increment    stop
      value  by this      when
      of     amount       >D
      A
A → must be a single variable (no expression)
                    (only length)

B
C    → numbers, variables or expressions
D
```

```
1.1 FOR X = 1, 2, 7; TYPE % 2, X, !

*GO
     = 1
     = 3          If C is omitted      computations done in floating point
     = 5          increment assumed    arithmetic-hence name your format
     = 7          to be 1
1.1 FOR X = 1, 7; TYPE % 2, X, !
```

```
1.1 FOR I - 1, 6; ASK ? A [I] ? , !

*GO
   A [I]:
   A [I]:
   A [I]:
   A [I]:
   A [I]:
   A [I]:
```

```
1.1  FOR X = 1, .1, 2; TYPE X,!
                  try this
```

	NEW FOCAL	OLD FOCAL
YES	$.25*10^{21}$	9000
NO	155	155
FIND	7044	7044
HELP	$.799993*10^{137}$	.800011E+121

Program to make table of squares, square roots

```
1.1   SET B=1; SET E=5
1.2   TYPE "NUMBER          SQUARE                SQUARE ROOT",!!
1.3   FOR N=B, E; TYPE !,%3, N, % 8, N↑2, %8.05, FSOT (N)
1.4   ASK " ␣ ␣ MORE?", R; IF (R-155) 1.5, 1.6, 1.5
1.5   SET G=B+5; SET E=E+5; GOTO 1.3
1.6   QUIT
```

This concept can be used to determine the number of terms in a series such as
$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\text{----}$

GO?
One letter rather than full command

ASSIGNMENT

*1. Write a program which will give the factorials up to N, factorial where N is supplied at execution. (This should differ from #7 on problem set V in that you can use DO and FOR)

*2. Write a program which will compute and type the amount of money you will have in the bank after 5 years if you initially deposit $1000. Allow the interest rate to vary from 4% to 6% in steps of 1/4 %. Also allow the compounding to be done semi-annually, quarterly and daily.
The formula is $A = P(1 + \frac{R}{N})^{YN}$

Y = # of years
N = # of times compounded per year
R = rate of interest
P = initial deposit

*3. In a right triangle the legs are A and B and the hypotenuse is C. If M > N, set $A = 2MN$, $B = M^2-N^2$ and $C = M^2+ N^2$. All such triples are called PYTHAGOREAN TRIPLES (integer values satisfying the pythagorean theorem, viz $A^2+B^2= C^2$). Write a program which will give the triples generated for M going from 2 to 10 and N assuming values from 1 to M-1 in each case. (45 of them.)

*4. Write a program to approximate e using the series
$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \text{---} + \frac{1}{n!} + \text{---}$

*5. Write a program which will ask for two sides and the included angle of a triangle (in degrees) and will type the length of the third side.
Use the law of cosines $a^2=b^2+c^2-2bcCosA$

RANDOM NUMBERS

FRAN ( ) gives a number between ±1
anything in here is ignored

$0 \leq$ FABS (FRAN()) $\leq$

$0 \leq$ FABS (FRAN90) * N $\leq$ N

FITR (FABS (FRAN())*n) yeilds the integers 0, 1, 2, ---, N

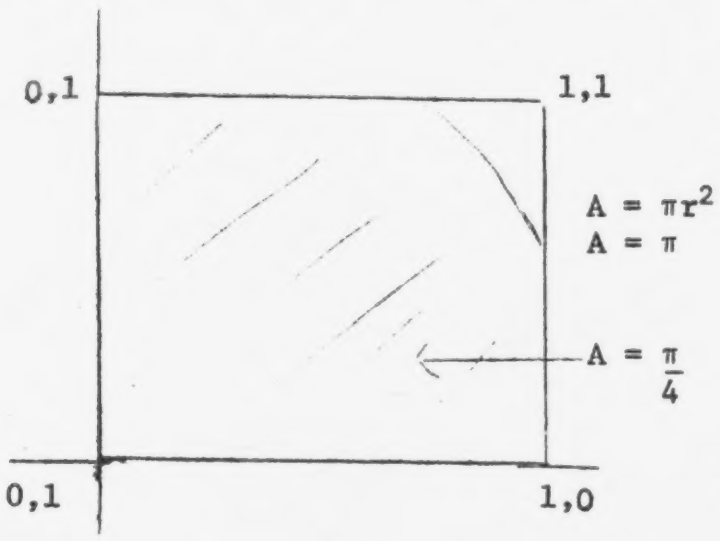
FITR (FABS (FRAN()) *10) yields the integers 0, 1, 2, ---, 10

FITR (FABS(FRAN())*5) + 1 yields 1, 2, 3, 4, 5, 6

FUN + PRACTICAL (simulation)

I. Fun
   a. dice
   b. baseball
   c. toss of a penny
      etc.

II. PRACTICAL
   a. Approximate π
   b. gas laws in chemistry



$A = \pi r^2$
$A = \pi$

$A = \frac{\pi}{4}$

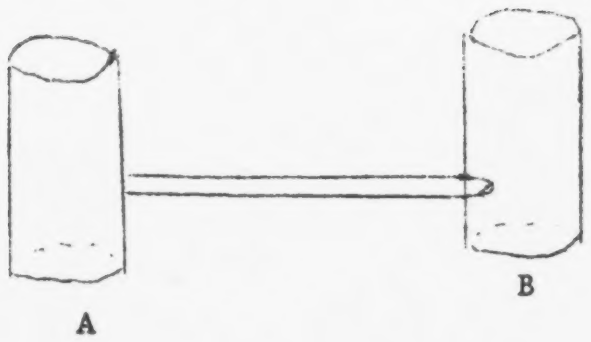
Asq = 1

$\frac{\text{No. of Hits}}{\text{No. of throws}} = \frac{\pi/4}{1}$

$\therefore \pi \approx 4 * (\text{HITS}) / (\text{THROWS})$

$X^2+Y^2 = 1$

If (X^2+Y^2-1) hit, hit, miss



100 balls no balls

```
1.1 SET A = 100; SET B = 0; FOR I = 1, 100; SET B[I] = 1      1;10
1.2 FOR C = 0, 10, 300; TYPE % 4.0, C, A, B,!; FOR I = C+1, C+10; DO2
```

```
2.1 SET X=FITR(FABS(FRAN())*99)+1; SET B[X]=-B[X]; IF (B[X]-1) 2.3, 2.2, 2.3
2.2  SET B=B-1; SET A=A+1; RETURN
2.3 SET B=B+1; SET A=A-1
```

APPENDIX VI

ASSIGNMENTS

The following may be assigned in either the BASIC or the FOCAL sections: (other problems are contained in each of the chapters)

Write a program to:

1. Compute the arc sin.

2. Compute the arc consine.

3. Round off numbers with decimal parts to the nearest whole number.

4. Average 5 numbers (in BASIC use the DATA statement)

5. Ask for and average 5 numbers.

6. Get a random digit. (uses INT and RND in BASIC)

7. Round off numbers to the nearest tenth.

8. Round off numbers to the nearest hundredth.

9. Round off numbers to the nearest thousandth.

10. Give integer random numbers from 5 to 24 inclusive.

11. Give X integer random numbers from B on.

12. List the squares from 1 to 20.

13. List a number and its cube from 8 to 15.

ASSIGNMENTS (cont.)

14. List a number and its square root from 7 to 23.

15. Print the values of X and Y if $Y = 3X^2 + 2X - 5$
and XE { -4, -3.5, -3, -2.5, ---, 2, 2.5, 3}

16. Convert degrees to radians.

17. Convert radians to degrees

18. Find the area of a circle when the radius is asked for at the time of execution.

19. Compute the values of the sine function for angles from 0^0 to 360^0 in steps of 30^0

20. Find the circumference of a circle when the radius is asked for at the time of execution.

21. Approximate π by using inscribed and circumscribed polygons.

22. Compute N! for an N supplied at time of execution.

23. Compute 1! thru N! for an N supplied at time of execution.

24. Compute N pythagorean triplets.

25. Approximate e using the series
$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \text{---}$

ASSIGNMENTS (Cont.)

26. Approximate sin X for an X supplied at time of execution using the series
$$\text{Sin } X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \frac{X^9}{9!} - \text{---}$$

27. Approximate cos X for an X supplied at time of execution using the series
$$\text{Cos } X = 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} + \text{----}$$

28. Find the greatest common divisor (G.C.D.) of two numbers.

29. Find the least common multiple (L.C.M.) of two numbers.

30. Ask for a number and print out whether it is prime or composite.

31. List the prime numbers up to N.

32. Convert from natural logarithms to common logarithms.

33. Convert from common logarithms to natural logarithms.

34. Convert from centigrade to fahrenheit.

35. Convert from fahrenheit to centigrade

36. Print the first N Fibonacci numbers.

37. Identify the type of conic section when A, B, C, D and E are supplied in the equation $AX^2 + BY^2 + CX + DY + E = 0$

ASSIGNMENTS (Cont.)

38. Find the maximum or minimum of the parabola $Y + AX^2 + BX + C$

39. Play TIC TAC TOE (use subscripted variables)

40. Approximate roots using Newton's tangent method.

41. Find the missing parts of a triangle if:
    a. Two angles and a side are given
    b. Two sides and the angle opposite one of them are given
    c. Two sides and the included angle are given.

42. Approximate the area under a curve using the trapezoidal rule.

43. Compute nPr

44. Compute nCr

45. Give the coefficients of the binomial expansion for a given integral exponent.

46. Convert a base 10 number to base 8.

47. Compute the mean and standard deviation of a set of N Scores.

$$S.D. = \sqrt{\frac{\sum_{L=1}^{N} (xi - M)^2}{N}} \quad \text{where } M = \text{mean}$$

48. Add N consecutive odd numbers beginning at 1.

ASSIGNMENTS (Cont.)

49. Solve a system of N linear equations with N. unknowns.
(Hint: use Gaussian elimination method)

50. Approximate the area under a curve between X = A and X = B using rectangles.

51. Play a game of dice. In the game of dice, a player bets even money on his chance of winning when he throws the two dice. If he throws 7 or 11 he wins immediately. If he throws 2, 3, or 12 he loses immediately. For all other numbers (i.e. 4, 5, 6, 8, 9, 10)he must continue to throw until either that number, called the point, appears again (and he wins) or the number 7 appears (and he loses).

52. Compute the limiting value of

$$1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ddots}}}}}}}$$

Your answer should be $\sqrt{2}$

53. Approximate π using the series:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \cdots\right)$$

54. Compute the area of a circle using random numbers.

55. Give the coefficients of a quadratic equation if the two roots are supplied.

ASSIGNMENTS (Cont.)

56. Give the values of A, B, and C in the parabola $Y = AX^2 + BX = C$ if three points on the parabola are supplied.

57. Give the area of a triangle if the lengths of the three sides are supplied.

58. Compute the weekly payroll for a company employing 10 men. Consider state and federal income tax, social security, hospital insurance. Do not assume all make the same salary nor have the same percentage of taxes withheld. Print out the payroll for six weeks and the total of each column for the six week period.

59. Determine if a point supplied at execution lies on, in, or outside the circle $X^2 + Y^2 = R^2$ where R is also supplied at execution.

60. Determine if a point supplied at execution satisfies the inequality $Ax + By > C$ where A, B and C are also supplied at execution.

61. Determine 10 points which satisfy the inequality $AX + BY \leq C$ where A, B and C are supplied at execution. Use random numbers and print each point with a "yes" or "no" until 10 "yes" answers have appeared.

62. Determine the nature of the roots of a quadratic equation using the discriminant.

63. Determine the distance between two points.

64. Determine the distance from a point to a line.

65. Find the area of any regular polygon.

66. Find the sum of a geometric progression.

ASSIGNMENTS (cont.)

67. Approximate e using $\left(1 + \frac{1}{n}\right)^n$

68. Determine if a natural number is even or odd.

69. Convert a number from base N to base 10.

70. Calculate the solution to the birthday problem. i.e. how many people are necessary to have a probability greater than 1/2 that two of them have the same birthday?

71. Convert a number from base 10 to any base.

72. Demonstrate that the set of rational numbers is dense.

73. Show $\lim_{x \to 0} \frac{\sin x}{x} = 1$

74. Simulate a baseball game.

75. Find the sum of the first N factorials.

76. Ask for a list of N numbers (any real numbers) and have the computer print out those that are between 10 and 17 inclusive or between 47 and 61 (not inclusive)

ASSIGNMENTS (cont.)

77. Ask for the Mathematics, Science, Foreign Language, English and Social Studies grades of N students. If a student's average (Ave) is 90 - 100 inclusive put him on a list labeled HIGH HONORS. If his average (Ave) is $80 \leq$ Ave < 90 put him on a list labeled HONORS. If the average is below 60 put him on a list labeled FAILURES.

78. Determine all four digit numbers which are perfect squares and each of whose digits is even.

79. Compute $\sum_{i=1}^{N} i^2 - i$

80. Compute a table of values of e^x for values of x between -1.00 and 1.00 in steps of .01 using
$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} = \frac{x^3}{3!} + \text{---} + \frac{x^n}{N!} = \text{---}$

81. Find the sum of the squares of the first N natural numbers showing each partial sum.

82. Find the least common multiple of N natural numbers.

83. Convert ft/sec to mi/hr

84. Find the magnitude and direction of the resultant of two vectors.

END